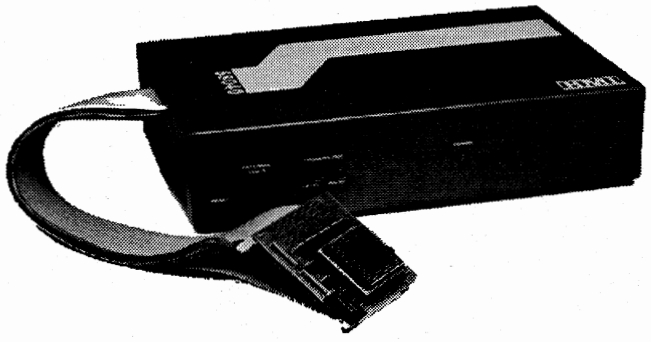
SourceGate™ II

Emulator Reference Guide

for the

HMI-200-CPU32



Revision 1.1.0

June 22, 1994

Huntsville Microsystems, Inc. © 1994

The contents of this document are believed to be accurate. However, Huntsville Microsystems, Inc. assumes no responsibility for the use of this document and issues no warranties, implied or otherwise, on the contents herein. Furthermore, Huntsville Microsystems, Inc. reserves the right to revise this publication and to make changes to the contents, without obligation to notify any persons of such revisions or changes.

Copyright © 1994 by Huntsville Microsystems, Inc.

SourceGate is a registered trademark of Huntsville Microsystems, Inc.
Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation.
Solaris and OpenWindows are trademarks of Sun Microsystems, Inc.

This document was created entirely with AmiPro, which is a trademark of Lotus Development Corporation.

TABLE OF CONTENTS

Chapter 1 - Getting Started 1

Emulator To Computer Connection 1

Emulator To Target System Connection 1

Starting SourceGate II 1

Chapter 2 - The CPU32 Interface 2

Defining The Emulator's Configuration Window 2

Terminal Type Configuration Windows 2

Terminal Type Configuration Window Fields 4

Clock Field 4

Target System: Reset Field 4

Target System: Br,Halt,Berr,Irqs Field 4

DSACK Source Is Field 4

Default Port Width Field 5

Register Reset Values Fields 5

CPU32 Terminal Type Configuration Window Memory Map Fields 5

E and T Fields 5

Activator Fields 5

Base Field 7

FC2-0 Field 7

Size Field 7

Width Field 8

RO and RW Fields 8

RAM Used/Left Fields 8

SR Used/Left Fields 8

Working With Shadow RAM 8

Graphical Configuration Windows 9

The CPU32 Source Code Interface Configuration 10

The CPU32 Command Window 11

Opening The Command Window 11

Command Window Buttons 12

TABLE OF CONTENTS (Cont.)

Entering Command Window Commands 12
Command Line Overrides 13
S (Supervisor Mode) 13
U (User Mode) 13
P (Program Mode) 13
D (Data Mode) 13
L (Long Word Mode) 13
W (Word Mode) 13
B (Byte Mode) 13
T (Target System) 13
R (Repeat) 14
Override Example 14
Using Symbols In The Command Window 15
Using Module Commands In The Command Window 16
:=A [module] Assembly Mode Command 16
:=C [module] Clear Symbol Table Command 16
:=D [module] Display Current Modes Command 16
:=M [module] Mixed Mode Command 16
:=S [module] Source Mode Command 17
Available CPU32 Commands 18
A (In-Line Assembly) Command 20
BM (Byte Mode) Command 21
C (Compare Memory Block) Command 22
CM (Show Status Of Control Board Memory) Command 24
CONFIG (Recall Factory Configuration) Command 25
CONFIGT (Recall Factory Configuration For Target) Command 26
CRL (Load Configuration Registers) Command 27
D (Dump Memory Block) Command 28
DM (Data Memory Mode) Command 29
DW (Write Line Active For Target Writes Only) Command 30
E/EN (Enter/Change Memory) Command 31

TABLE OF CONTENTS (Cont.)

EDT (Events Disabled For Temp. Breakpoints) Command 32
EET (Events Enabled For Temp. Breakpoints) Command 33
ET (Show Status For EDT/EET) Command 34
EW (Write Line Active For All Memory Writes) Command 35
EXP (Examine/Change Peripheral) Command 36
EXR (Examine/Change Peripheral Register) Command 37
F (Fill Memory Block) Command 38
G (Begin Real-Time Emulation) Command 39
HLD (Hex Listing Disabled) Command 40
HLE (Hex Listing Enabled) Command 41
I/IN (Input) Command 42
L (List Disassembled Code) Command 43
LM (Long Word Display Mode) Command 44
M (Move Memory Block) Command 45
MT/MTL (Test Block Of Memory) Command 46
O/ON (Output) Command 47
PD (Program Wait States For DSACK) Command 48
PM (Program Memory Mode) Command 49
RD/RDP/RDX (Read Object File Into Memory) Command 50
RP (Reset Processor, Continue Emulation) Command 52
RS (Reset Processor, Stop Emulation) Command 53
SH (Show Status For SHD/SHE) Command 54
SHD (Serial Handshake Disabled) Command 55
SHE (Serial Handshake Enabled) Command 56
SM (Supervisor Memory Mode) Command 57
SP (Stop Emulation) Command 58
SR/SRN (Search Block Of Memory) Command 59
SS (Single Step) Command 61
ST (Show Emulator Status) Command 64
UM (User Memory Mode) Command 65
VASM (View Assembly) Command 66

TABLE OF CONTENTS (Cont.)

VER (Display Firmware Version) Command 67
W/WX (Write Memory Contents To File) Command 68
WM (Word Display Mode) Command 70
X (Examine/Change Registers) Command 71
The CPU32 Event Windows 73
Opening The Event Windows 73
Event Window Hexadecimal Fields 73
Event Address Field 74
Event Address Range Field 74
Event Data Field 74
Event Data Range Field 74
Event Range Active Check Box 74
Event Range [INC]/]EXC[Check Boxes 74
Event Pass Count Field 75
Event Symbol Fields 75
Event Window Buttons 75
Event Window Binary Fields 76
Event Address Binary Window 76
Event Data Binary Window 77
Event External Binary Window 77
Event Status Binary Window 78
The CPU32 Trace Window 83
Opening The Trace Window 84
Trace Window Buttons 85
Choosing The Trace Buffer To View 86
Trace Display Modes 86
Disassembly 87
State 88
Source 90
Disassembly/State 91

TABLE OF CONTENTS (Cont.)

Disassembly/Source 93
Disassembly/State/Source 94
Using The Time Stamp Feature 96
Timing Off 96
Timing Relative 96
Timing Delta 98
Freeze Trace Operation 99
Freezing The Trace 99
Resuming The Trace 99
Searching The Trace 100
68330/68340 Trace Search Status Bits 102
68331/68332 Trace Search Status Bits 103
68333 Trace Search Status Bits 106
Trace Search Window Buttons 107
Printing The Trace 108
Print Trace Buffer Window Buttons 108
The CPU32 Register Window 109
Opening The Register Window 110
Modifying Register Values 110
Register Window Contents 110
Program Counter Register 110
Program Counter Description Field 110
Status Register 110
Supervisor Stack Pointer Register 111
User Stack Pointer Register 111
Vector Base Register 111
Data Registers D0-D7 111
Address Registers A0-A7 111
Alternate Function Code Registers 112

1 Getting Started

Your HMI-200-CPU32 emulator is designed to either run stand-alone or in a target system. SourceGate II is a source-level debugger that is the interface between you and your emulator. Since SourceGate II is a common user interface among all HMI products, most SourceGate II features and windows are the same for all emulators. However, certain areas such as the Configuration, Event, Trace, and Register windows along with the Command Window commands, are different for each emulator. **This manual will address those areas that are unique to each emulator.** For additional information on using the common SourceGate II features for your particular environment, reference one of the following manuals:

SourceGate™ II for Motif User's Guide
SourceGate™ II for Windows User's Guide
SourceGate™ II for OpenWindows/Solaris User's Guide

The rest of this manual is dedicated to describing the unique features of your particular emulator. Note that the window examples shown in this document were captured under Microsoft Windows and may not reflect the exact "look" of the windows shown in your particular environment. However, the window contents, and the operations that are performed, should be identical.

Emulator To Computer Connection

Before starting SourceGate II, you must connect a serial cable from a serial port of the computer to the "MAIN" port of the emulator. This should be a "straight-thru" type of cable. For best results, use the cable supplied with your emulator.

Emulator To Target System Connection

You don't need to have the emulator connected to a target system for stand-alone operation. However, if you wish to use your target system, you must connect the emulation pod to your target system. To do this, remove your cpu32 processor from its socket. Place the emulation pod in this open socket. If you are using the clip over or surface mount adapter, follow the instructions supplied with the adapter. Make sure you orient the pod correctly in the target socket or damage may result to the pod, target system, or both. Once this is done, you need to configure the emulator to use the desired target system signals. For additional information on how to do this, see ***Defining The Emulator's Configuration Window*** in Chapter 2.

Starting SourceGate II

Once the proper connections have been made, you need to start SourceGate II in order to communicate with the emulator. Reference the ***SourceGate II User's Guide*** for your operating environment for information on how to install and invoke SourceGate II.

2 The CPU32 Interface

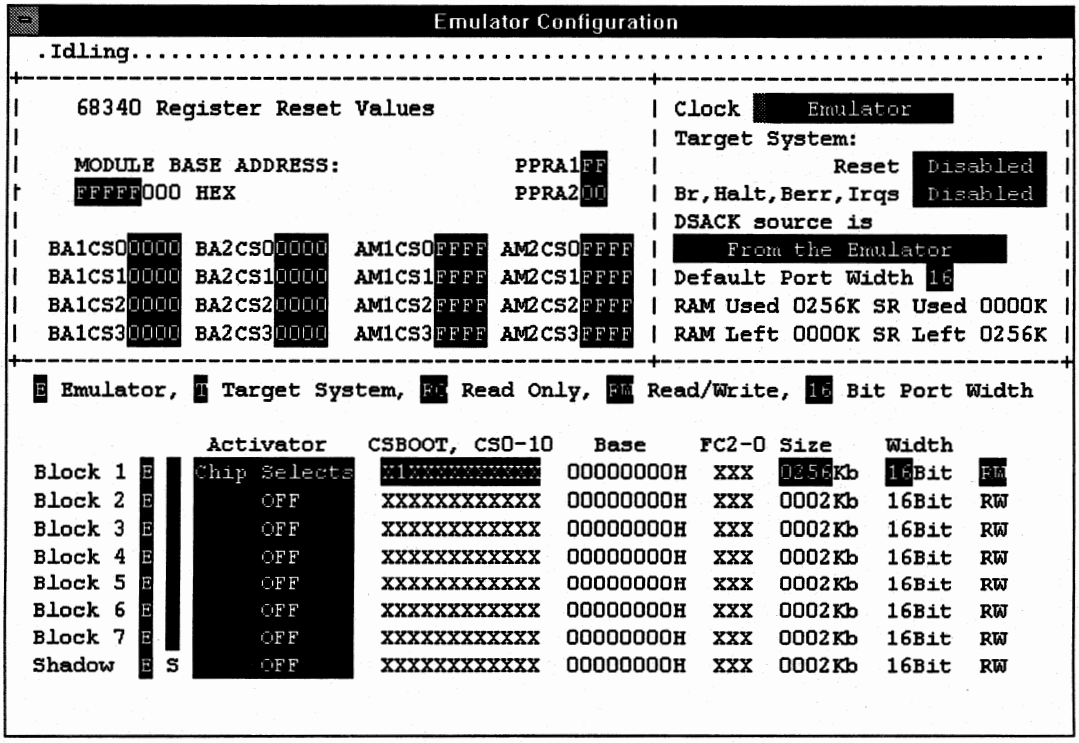
Since SourceGate II is a common user interface among all HMI products, most SourceGate II features and windows are the same for all emulators. However, certain areas such as the Configuration, Event, Trace, and Register windows along with the Command Window commands, are different for each emulator. This chapter will address these differences for the HMI-200-CPU32 emulator. The HMI-200-CPU32 emulator currently has pods available which allow you to support the following processors: 68330, 68331, 68332, 68333, and 68340. Any features unique to one of these processors will be outlined in the sections that follow.

Defining The Emulator's Configuration Window

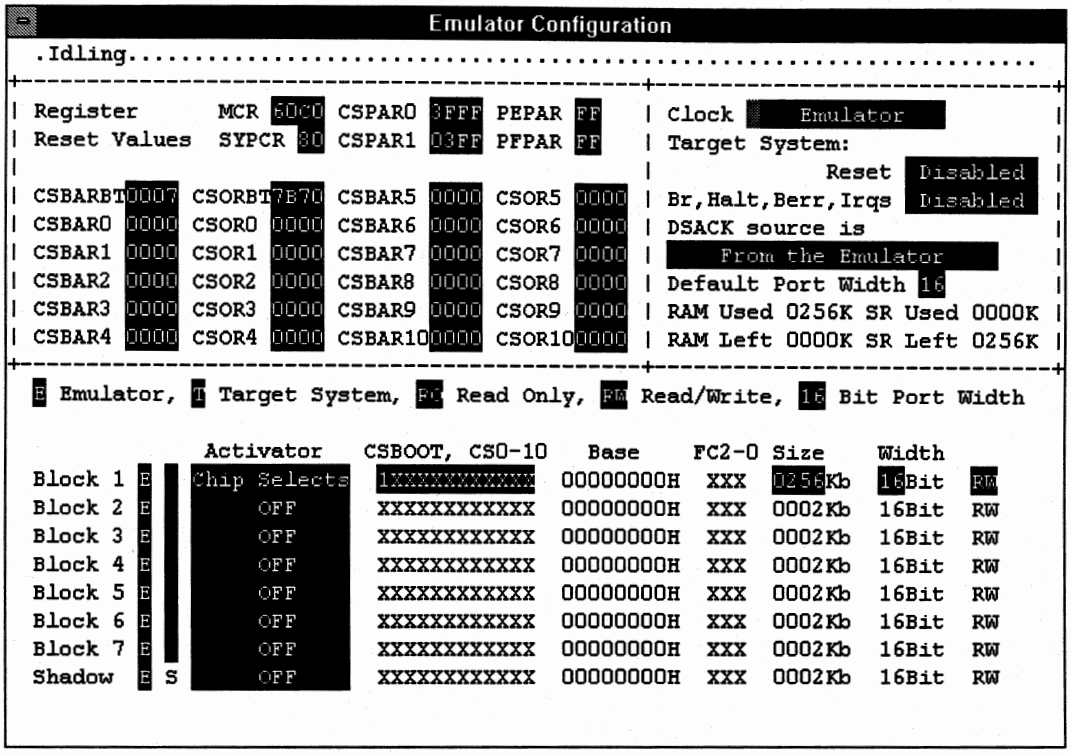
Each emulator has a Configuration window that allows you to set up the memory map and define how certain signals are handled. To open this window, choose 'Config/Memory'. There are two Configuration windows that may be shown depending on the version of the emulator being used. An example of each of these for the HMI-200-CPU32 is shown in the following sections.

Terminal Type Configuration Windows

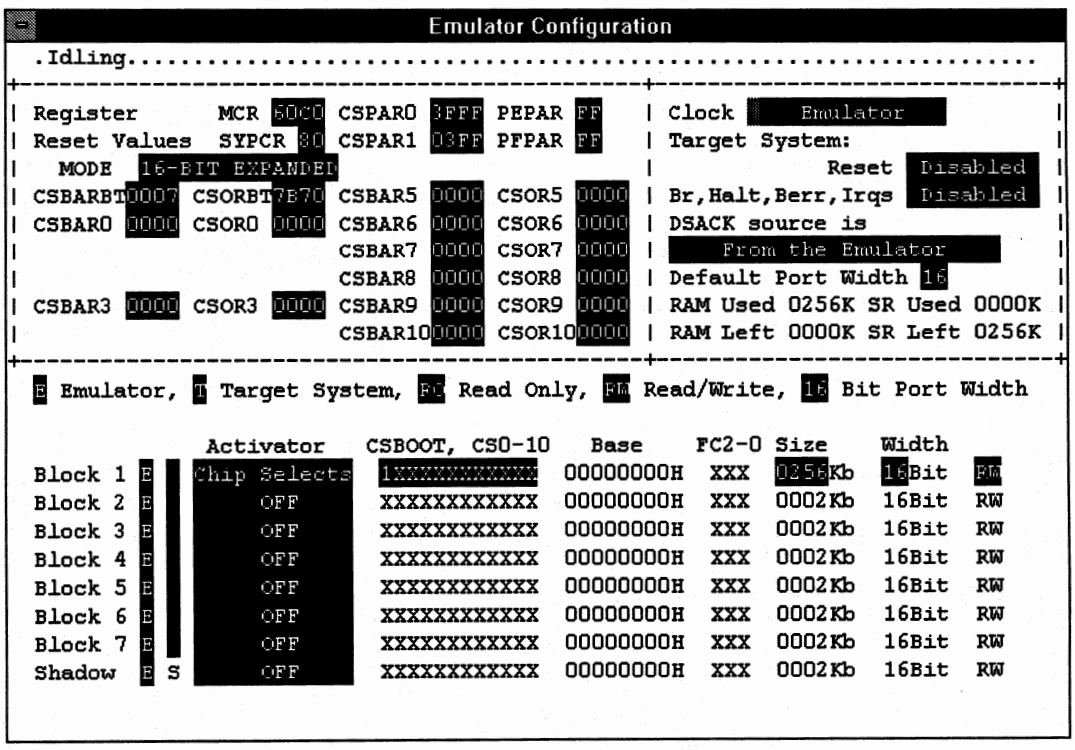
Some 68330 and 68340 emulators being used with SourceGate II will show a terminal type Configuration window with the following format:


Emulator Configuration
.Idling.....
68340 Register Reset Values
MODULE BASE ADDRESS:
FFFFF000 HEX
PPRA1FF
PPRA200
Clock Emulator
Target System:
Reset Disabled
Br,Halt,Berr,Irqs Disabled
DSACK source is
From the Emulator
Default Port Width 16
RAM Used 0256K SR Used 0000K
RAM Left 0000K SR Left 0256K
BA1CS0 0000 BA2CS0 0000 AM1CS0 FFFF AM2CS0 FFFF
BA1CS1 0000 BA2CS1 0000 AM1CS1 FFFF AM2CS1 FFFF
BA1CS2 0000 BA2CS2 0000 AM1CS2 FFFF AM2CS2 FFFF
BA1CS3 0000 BA2CS3 0000 AM1CS3 FFFF AM2CS3 FFFF
E Emulator, T Target System, RO Read Only, RW Read/Write, 16 Bit Port Width
Activator CSBOOT, CS0-10 Base FC2-0 Size Width
Block 1 E Chip Selects X1XXXXXXXXXX 00000000H XXX 0256Kb 16Bit RW
Block 2 E OFF XXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 3 E OFF XXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 4 E OFF XXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 5 E OFF XXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 6 E OFF XXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 7 E OFF XXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Shadow E S OFF XXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW

The 68331 and 68332 emulators will show the following terminal type Configuration window:


Emulator Configuration
.Idling.....
Register MCR 60C0 CSPAR0 3FFF PEPAR FF
Reset Values SYPCR 80 CSPAR1 03FF PFPAR FF
CSBARBT 0007 CSORBT 7B70 CSBAR5 0000 CSOR5 0000
CSBAR0 0000 CSOR0 0000 CSBAR6 0000 CSOR6 0000
CSBAR1 0000 CSOR1 0000 CSBAR7 0000 CSOR7 0000
CSBAR2 0000 CSOR2 0000 CSBAR8 0000 CSOR8 0000
CSBAR3 0000 CSOR3 0000 CSBAR9 0000 CSOR9 0000
CSBAR4 0000 CSOR4 0000 CSBAR10 0000 CSOR10 0000
Clock Emulator
Target System:
Reset Disabled
Br,Halt,Berr,Irqs Disabled
DSACK source is
From the Emulator
Default Port Width 16
RAM Used 0256K SR Used 0000K
RAM Left 0000K SR Left 0256K
E Emulator, T Target System, RO Read Only, RW Read/Write, 16 Bit Port Width
Activator CSBOOT, CS0-10 Base FC2-0 Size Width
Block 1 E Chip Selects 1XXXXXXXXXXXX 00000000H XXX 0256Kb 16Bit RW
Block 2 E OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 3 E OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 4 E OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 5 E OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 6 E OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 7 E OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Shadow E S OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW

The 68333 emulator will show the following terminal type Configuration window:


Emulator Configuration
.Idling.....
Register MCR 60C0 CSPAR0 3FFF PEPAR FF
Reset Values SYPCR 80 CSPAR1 03FF PFPAR FF
MODE 16-BIT EXPANDED
CSBARBT 0007 CSORBT 7B70 CSBAR5 0000 CSOR5 0000
CSBAR0 0000 CSOR0 0000 CSBAR6 0000 CSOR6 0000
CSBAR7 0000 CSOR7 0000
CSBAR8 0000 CSOR8 0000
CSBAR3 0000 CSOR3 0000 CSBAR9 0000 CSOR9 0000
CSBAR10 0000 CSOR10 0000
Clock Emulator
Target System:
Reset Disabled
Br,Halt,Berr,Irqs Disabled
DSACK source is
From the Emulator
Default Port Width 16
RAM Used 0256K SR Used 0000K
RAM Left 0000K SR Left 0256K
E Emulator, T Target System, RO Read Only, RW Read/Write, 16 Bit Port Width
Activator CSBOOT, CS0-10 Base FC2-0 Size Width
Block 1 E Chip Selects 1XXXXXXXXXXXX 00000000H XXX 0256Kb 16Bit RW
Block 2 E OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 3 E OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 4 E OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 5 E OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 6 E OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Block 7 E OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW
Shadow E S OFF XXXXXXXXXXXX 00000000H XXX 0002Kb 16Bit RW

Each of the fields shown in the preceeding windows are outlined in the sections that follow.

Terminal Type Configuration Window Fields

Many of the fields shown in these windows must be configured differently to work with specific target systems. Any field that is highlighted can be changed by moving the cursor to that field and hitting the space bar. Each field in these windows is described in the sections that follow:

Clock Field

The source of the clock can be the "Emulator" on-board clock or the "Target System" clock.

Target System: Reset Field

The "Reset" signal can be "Enabled" or "Disabled". "Enabled" means that this signal can influence the processor during emulation. "Disabled" means that this signal is gated off and cannot influence the processor. This signal should be "Disabled" when the emulator is not connected to a target system.

Target System: Br,Halt,Berr,Irqs Field

These target system control signals can be "Enabled" or "Disabled". "Enabled" means that these signals can influence the processor during emulation. "Disabled" means that these signals are gated off and cannot influence the processor. These signals should be disabled when the emulator is not connected to a target system.

DSACK Source Is Field

The CPU32 processor requires a DSACK signal to acknowledge all memory operations. This signal can come "From the Emulator", "From the Target System", or "According to Memory Map".

When running the emulator stand-alone without a target system, DSACK should be selected to come "From the Emulator". The "Default Port Width" for this emulator generated DSACK signal should be entered at this time. The choices for the default port width field are 8 and 16 bits. It should be noted the the "Width" field in the lower area of the menu can override the default port width when addresses in the defined block are accessed.

When plugged into a target system, DSACK can be selected to come "From the Target System" or "According to Memory Map". If "According to Memory Map" is chosen, then the DSACK signal will come from the emulator when the processor makes a memory access to emulator memory, or from the target system when the processor makes a memory access to target system memory.

Default Port Width Field

When running the emulator stand-alone without a target system, DSACK should be selected to come "From the Emulator". The "Default Port Width" for this emulator generated DSACK signal should be entered at this time. The choices for the default port width field are 8 and 16 bits. It should be noted the the "Width" field in the lower area of the menu can override the default port width when addresses in the defined block are accessed.

Register Reset Values Fields

The values normally programmed into the SIM registers should be defined here. Any time you perform a reset of the CPU32 processor, the SIM registers will be programmed with the values entered here. These registers should contain the same information as your startup code to avoid conflicts.

Note that there is also a "MODE" field in the 68333 Configuration window. This field determines what mode the chip is to be set in upon a reset. The choices are "8-Bit Expanded", "16-Bit Expanded", and "Single Chip". This field should be configured to reflect how your hardware sets the mode of the processor when a reset operation is performed.

CPU32 Terminal Type Configuration Window Memory Map Fields

The CPU32 emulator comes standard with 256K bytes of emulation memory and 256K bytes of Shadow RAM (Shadow RAM may only be available on newer units). The bottom of the screen will display seven blocks where the emulation memory can be mapped and one block where the Shadow RAM is mapped. These blocks can be configured to define blocks used as actual memory or for the definitions of chip selects. When any of these blocks are modified, the emulator immediately issues a reset to the processor in order to program the new conditions and the message "PROGRAMMING" is displayed. If there are conflicts between what is defined here and what is programmed in the SIM registers, the message "Possible invalid selection(s) in Config. Menu." will be displayed.

Each of the memory map fields is described in the following sections. Any processor specific differences will be noted.

E and T Fields

To the right of each block number in the memory table is a toggle field which will specify 'E' or memory defined in the size, base, and space fields will be referenced in the emulator. If the 'T' is selected, then the block of memory will be referenced in the Target System. All blocks can be changed to emulator or target by placing the cursor on either the "E" or "T" fields and hitting the space bar.

Activator Fields

The "Activator" field toggles between "Chip Selects", "Base & FC2-0", and "OFF". The activator field determines what combination of parameters that must occur before the defined block of memory will be activated. Each of these activators is described as follows:

Chip Selects: (For 68330/68340)
If "Chip Selects" is chosen as the activator, the "CSBOOT, CS0-10" field either shows a "1" or a "X" for each Chip Select. For these processors, only CS0-CS3 are valid and can be changed. All other chip selects will remain disabled with the "X" value shown for that position. A "1" means that Chip Select will activate the defined block. A "X" means that Chip Select is inactive. For example, if Block 1 had the following definition:

Blk 1 E Chip Selects X11X1XXXXXXX

then this block would be activated anytime CS0, CS1, or CS3 became active. It is important to note that the Chip Select Option registers define all of the aspects of the memory block (Read-Only, Block Size, Width, Base Address, etc.) when using "Chip Selects" as the activator.

Chip Selects: (For 68331/68332)
If "Chip Selects" is chosen as the activator, the "CSBOOT, CS0-10" field either shows a "1" or a "X" for each Chip Select. A "1" means that Chip Select will activate the defined block. A "X" means that Chip Select is inactive. For example, if Block 1 had the following definition:

Blk 1 E Chip Selects 1X1X1XXXXX11

then this block would be activated anytime CSBOOT, CS1, CS3, CS9, or CS10 became active. It is important to note that the Chip Select Option registers define all of the aspects of the memory block (Read-Only, Block Size, Width, Base Address, etc.) when using "Chip Selects" as the activator.

Chip Selects: (For 68333)
If "Chip Selects" is chosen as the activator, the "CSBOOT, CS0-10" field either shows a "1" or a "X" for each Chip Select. For this processor, CS1', CS2', and CS4' will remain disabled with the "X" value shown for that position. A "1" means that Chip Select will activate the defined block. A "X" means that Chip Select is inactive. For example, if Block 1 had the following definition:

Blk 1 E Chip Selects 11XX1XXXXX11

then this block would be activated anytime CSBOOT', CS0', CS3', CS9', or CS10' became active. It is important to note that the Chip Select Option registers define all of the aspects of the memory block (Read-Only, Block Size, Width, Base Address, etc.) when using "Chip Selects" as the activator.

Base & FC0-2:
If "Base & FC0-2" is chosen as the activator, then a combination of the "Base", "FC2-0", "Size", "Width", and "RW/RO" fields should be used to define the block activator. The "FC2-0" field can be any combination of "0", "1", or "X" where "X" is the "Don't Care" condition.

OFF:
When a block is turned "OFF", the emulator will disable that block in the defined memory area (emulator or target) unless the same area is enabled under another block.

For example, let two blocks be defined as follows:

Blk1 E Base&FC2-0 40000 111 0002Kb 16Bit RW

Blk2 E OFF 40000 111 0002Kb 16Bit RW

then memory at addresses 40000H - 407FFH can be addressed in the emulator since the "Base&FC2-0" activator is overlaying the same block that is turned "OFF".

If however, the following blocks were defined:

Blk1 E Base&FC2-0 80000 111 0002Kb 16Bit RW

Blk2 E OFF 40000 111 0002Kb 16Bit RW

then any accesses to addresses 40000H - 407FFH would be referenced in target memory since the block at these addresses in the emulator is turned "OFF".

Base Field
The "Base" field defines the base address of the defined block and is used along with the "Width" field to define the size of the block. This field is only valid when the "Base & FC2-0" option is chosen for the activator, and is invalid for all other options.

NOTE: This field should not be confused with the chip select base registers that are internal to the processor.

FC2-0 Field
The "FC2-0" field allows the user to define the function code pattern which will activate the defined block. This field is only valid when the "Base & FC2-0" option is chosen for the activator, and is invalid for all other options.

Size Field
The "Size" field defines the size of the block to be defined. This field can be changed by moving the cursor to the field and toggling the space bar. If the "Base & FC2-0" activator is used, then this field tells the emulator how much memory is defined for the particular block starting at the "Base" field shown in this menu. If the "Chip Selects" activator is used, then this field tells the emulator how much memory is defined for the particular block starting at the base address for the defined chip select. This value will override the block size defined in the chip select option register.

The options for this field are: 2K, 4K, 8K, 16K, 32K, 64K, 128K, 256K, 512K, and 1024K.

Width Field

The "Width" field defines the width of the DSACK signal returned from the block. If the "Chip Selects" activator is used, this field will override the width defined in the chip select option register. The DSACK width defined in this field will override the default port width value when a defined block is accessed. This field can be changed by moving the cursor to the field and toggling the space bar.

RO and RW Fields

This field will toggle between 'RW' and 'RO'. This field is used for write protecting Memory in the emulator. Selecting an 'RW' will allow the associated block to be written to during emulation. Selecting an 'RO' will prevent the associated block from being written to during emulation or single stepping. If the "Chip Selects" activator is used, selecting "RO" in this field will cause the block to be read- only, even if the chip select option register defines it to be read/write. However, choosing "RW" will not override the chip select option register if that register defines the block to be read-only. All blocks can be changed to "Read/Write" or "Read- Only" by placing the cursor on either the "RW" or "RO" fields and hitting the space bar.

RAM Used/Left Fields

The total memory allocated in defined emulator memory blocks may not exceed the available emulation memory. "RAM Used" and "RAM Left" is displayed in the menu for quick reference. If the block size specified exceeds the amount of unused emulation memory, the message "OUT OF MEMORY" will be displayed.

SR Used/Left Fields

The total Shadow RAM memory allocated in defined emulator memory blocks may not exceed the available Shadow RAM memory. "SR Used" and "SR Left" is displayed in the menu for quick reference. If the block size specified exceeds the amount of unused Shadow RAM memory, the message "OUT OF SHADOW MEMORY" will be displayed.

Working With Shadow RAM

If your unit supports Shadow RAM, instead of the "Block 8" field in the Configuration window, you will be shown a field labeled "Shadow". This 256KB of Shadow RAM can be used as an additional block of emulation memory, or it can be used to "shadow" other blocks of emulation memory. This shadowing ability allows you to view, in real time, the contents of any of the 256KB of Shadow RAM using the Watch In Real Time window.

To shadow a defined memory block, move the cursor to the field just to the left of the "Activator" field for that block and press the space bar. This will toggle shadowing on and off. When a block is being shadowed, "S" will appear in this field. Notice that the "SR Used" and "SR Left" fields will be changed to reflect the size of the block being shadowed.

The block labeled "Shadow" can be used as an additional block of emulation memory by mapping it like any other memory block using the "Activator", "Size", etc. fields to determine when the block is active. Any Shadow RAM that is left after shadowing specific blocks can be used for this purpose, providing you with emulation memory beyond the standard 256KB supplied with the unit.

The Watch In Real Time window allows you to watch memory locations that reside in Shadow RAM defined in the Configuration window. This window differs from the standard Watch window in that the variables shown are updated dynamically in real-time during emulation. Entering variables into this window is accomplished in the same manner as for the standard Watch window. For additional information on how to enter variables, see ***Using The Watch In Real Time Window*** in your ***SourceGate II User's Guide***.

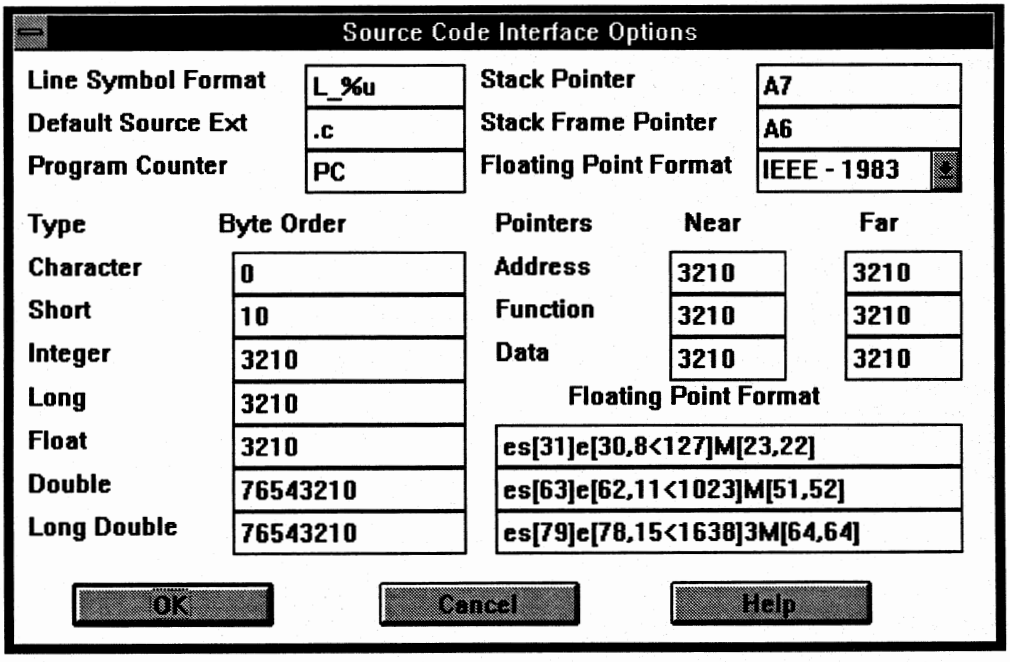
Note: In order to watch variables in the Watch In Real Time window, the address of the variable being watched must reside in a defined Shadow RAM block in the Configuration window. If a variable is entered that does not reside in this block, the message "[space]addr DUAL PORT NOT MAPPED" will be displayed., where "space" is the address space (supervisor program, user data, etc.) and "addr" is the first address encountered where the Shadow RAM is not mapped. The remedy for this situation is to move a Shadow RAM block in the Configuration window to where it covers the area you wish to watch.

Graphical Configuration Windows

Future versions of this emulator may have graphical Configuration windows that contain the same information that was discussed in the previous sections. Currently, there are no graphical Configuration windows available for this emulator.

The CPU32 Source Code Interface Configuration

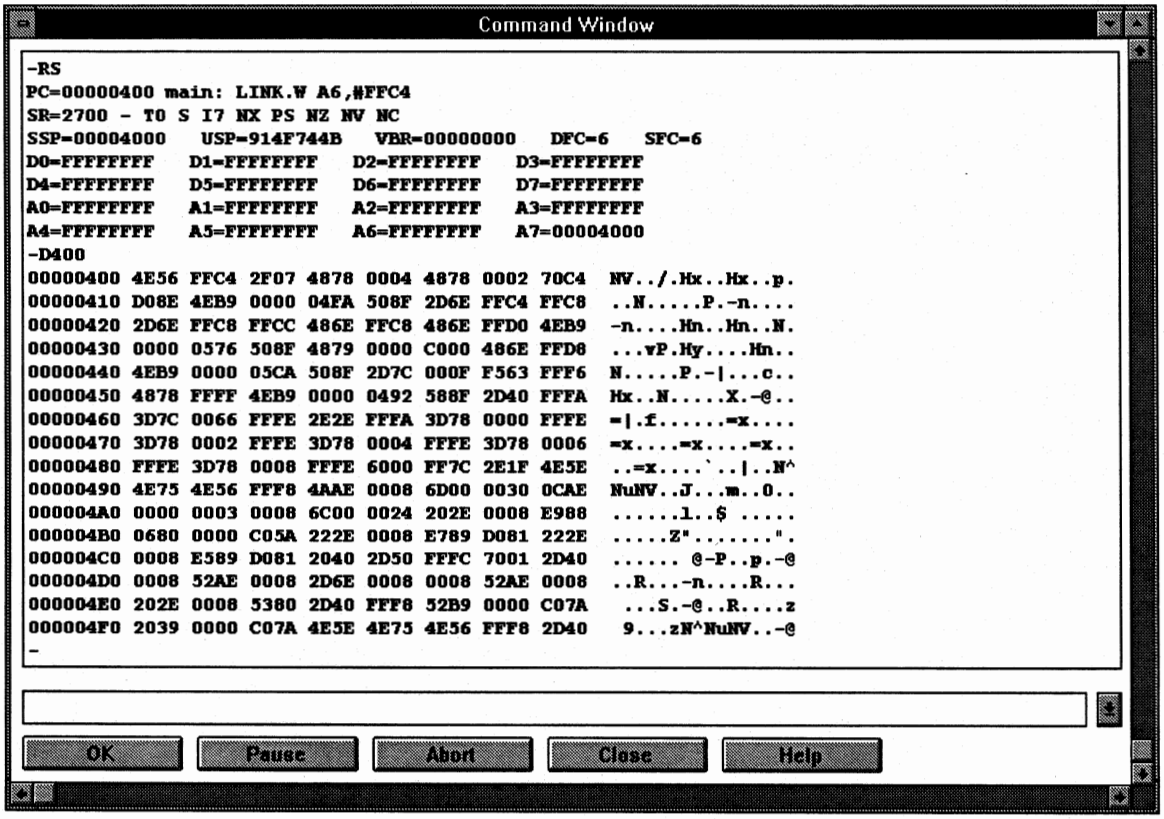
The Source Code Interface window is used to configure SourceGate II to work with almost any compiler. You can define several compiler specific parameters here such as the registers to be used for stack and frame pointers, the line symbol format used by the compiler, etc. To open this window, choose 'Config/Source Code Interface'. The following window shows the default setup for the HMI-200-CPU32 emulator:



You can change these setups to match your compiler if needed. This information is saved in the SGATE.INI file as soon as the changes are made.

The CPU32 Command Window

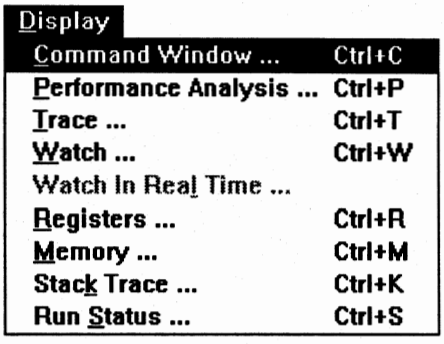
The Command window is the window that you will use to enter commands that affect the emulator. Emulator responses to these commands will also be shown in this window. The Command window has the following format:



An overview of the operation of the Command window follows.

Opening The Command Window

To open the Command window, choose 'Display/Command Window':



An alternate method of opening the Command window is to use the accelerator key combination Ctrl+C.

Command Window Buttons

A description of the functions of the buttons available in this window are as follows:

Ok

The **Ok** button always executes the command displayed in the command entry field.

Pause

The **Pause** button stops the information from scrolling on the screen. After the **Pause** button is pressed, the button will display **Resume**. Pressing the **Resume** button will allow the scrolling to continue.

Abort

The **Abort** button will abort the command that is currently executing.

Close

The **Close** button will close the Command window.

Help

The **Help** button will invoke Command window help.

Scroll Box

The **Scroll Box** button will allow you to scroll through any previously entered commands.

Entering Command Window Commands

All commands are entered in the command entry field located in the Command Window. Once a command is entered, the ENTER key or OK button will initiate execution of the command. Previously entered commands can be recalled by pressing the scroll box icon located next to the command entry field. A previously entered command can be highlighted using the mouse or the arrow keys. The highlighted command can be then be executed by pressing the ENTER key or OK button.

In the Command Window, the command abbreviation and the first parameter are not required to be separated, but may be separated by one or more spaces. If multiple parameter fields are used, they must be separated by a recognized delimiter. Allowed delimiters include a comma, a single space, or multiple spaces. If a comma is used as the delimiter, it may be preceded and followed by one or more spaces. This allows you some flexibility in entering the commands in a form that is comfortable and also allows for minor typing errors.

Command Line Overrides

An override is an optional command parameter that allows you to temporarily override the established global (default) parameter. Overrides are entered on the command line in brackets or parenthesis. More than one override may be specified in the brackets and several may be entered immediately adjacent to each other without separating spaces or delimiters.

Command line overrides do not apply to all commands. They are represented in the command format as a square brackets enclosing the override codes that apply to the command.

S (Supervisor Mode)

Memory addresses given in the command are assumed to be references into either the Supervisor Data or Supervisor Program memory partition. Which one is used will be determined by other overrides or global parameters.

U (User Mode)

Memory addresses given in the command are assumed to be references into either the User Program or User Data memory partition. Which one is used will be determined by other overrides or global parameters.

P (Program Mode)

Memory addresses given in the command are assumed to be references into either the User Program or Supervisor Program memory partition. Which one is used will be determined by other overrides or global parameters.

D (Data Mode)

Memory addresses given in the command are assumed to be references into either the User Data or Supervisor Data memory partition. Which one is used will be determined by other overrides or global parameters.

L (Long Word Mode)

The optional data size of input parameter or output (display) values is selected to be "Long Word".

W (Word Mode)

The optional data size of input parameter or output (display) values is selected to be "Word".

B (Byte Mode)

The optional data size of input parameter or output (display) values is selected to be "Byte".

T (Target System)

Memory will be accessed in the Target System's memory space overriding the memory map as defined in the configuration menu.

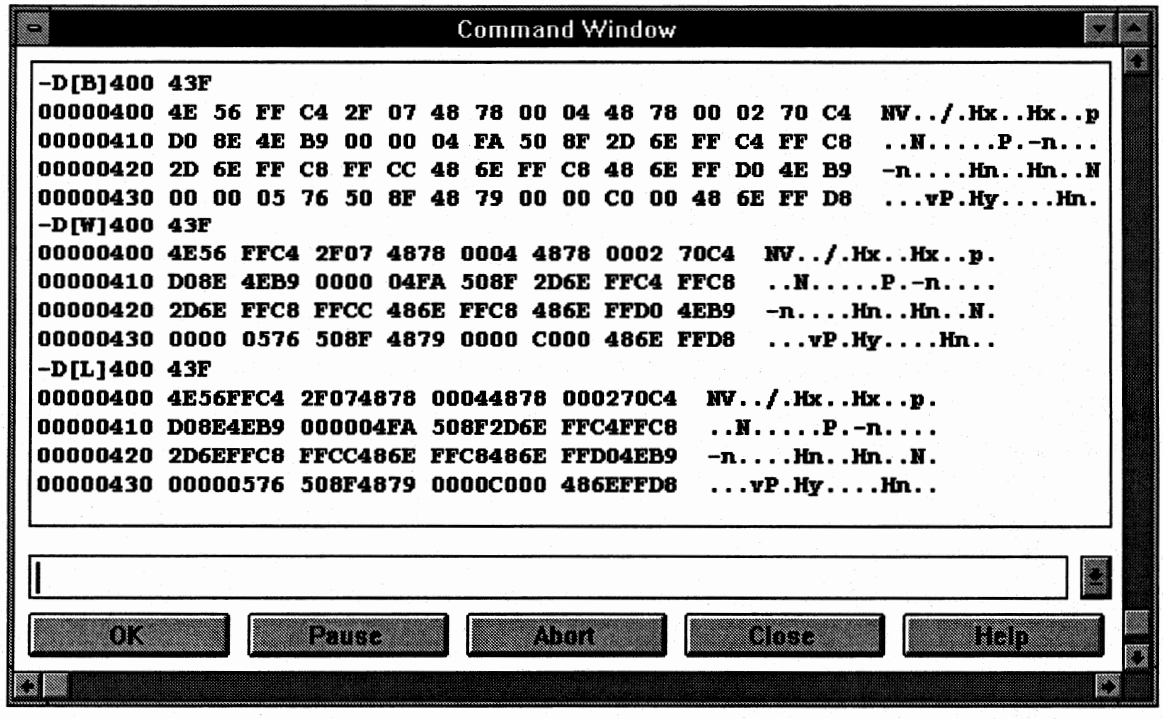
R (Repeat)
Rd or Rx
This override will re-execute the current command in three optional formats.

If an R is entered alone, then the command will re-execute each time ENTER is pressed until a period followed by ENTER is pressed.

If an R is entered followed by a number, then the command will re-execute 'd' times.

If an R is entered followed by an X, then the command will be re-executed continually until ENTER is pressed.

Override Example
The following is an example of using the "B", "W", and "L" overrides with the "D" or "Dump" command. Overrides are used similarly with other commands:



Using Symbols In The Command Window

Symbols can be entered in lieu of addresses or data values in Command window commands by typing a colon (:) followed by the symbol name. The value associated with the symbol name will be substituted in the command. Also, a number can be added to or subtracted from a symbol value by following the symbol with "+value" or "-value", respectively, where "value" is the number to be added.

The ':' command is used for symbol table display and editing. The 'symbol', '=', and 'value' portions of this command are all optional. This command comes in many different formats. Each of these is discussed in the sections that follow.

:

':' given by itself on a line causes SourceGate to display the entire contents of its internal symbol table.

:symbol

':' followed by a symbol will cause SourceGate to search for and display the first occurrence of the named symbol in the internal symbol table.

:symbol*

':' followed by a symbol with a wildcard character '*' will cause SourceGate to search for and display all occurrences of the symbol name including wildcards. For example, if ":test*" is entered, SourceGate will display all symbols of this type such as "test", "test1", "testit", etc.

:module/symbol

':' followed by a module name, followed by a '/', followed by a symbol name will cause SourceGate to search for the named symbol in or following the named module in the symbol table. If the symbol is found, it will be displayed.

:symbol=value
:module/symbol=value

These ':' commands perform symbol lookup as outlined above, but when the named symbol is found, the old value of the symbol is discarded, and the given value substituted for it.

'value' may be given as a hexadecimal number, or may be given as the name of another symbol, preceded by a colon. If the named symbol does not exist it will be created. If 'module' is given, the symbol will be created in the named module, otherwise a dummy module name of "USER_DEFINED_MODULE" is assumed.

Using Module Commands In The Command Window

Module commands can be performed while in the Command window. These commands influence the information that is shown in the Command window. The following commands are used to enable a particular level of debugging on a module or globally.

:=A [module] Assembly Mode
:=C [module] Clear Symbol Table
:=D [module] Display Current Modes
:=M [module] Mixed Mode
:=S [module] Source Mode

An explanation of each of these commands is given in the sections that follow.

:=A [module] Assembly Mode Command

The ':=A' command sets the named module (or all modules if none is named) to Assembly Level Debug mode. This is the default mode for all modules when SourceGate is started. While in this mode, only assembly language will be shown in the output of any Command window command, and all source code information will be suppressed.

:=C [module] Clear Symbol Table Command

The ':=C' command clears all symbols from the internal symbol table for the named module (or all modules if none is named). The symbol information for the cleared module must be read back into the internal symbol file in order to perform symbolic/source debugging on that module.

:=D [module] Display Current Modes Command

The ':=D' command displays source debug information for the named module (or all modules if none is named). Source debug information consists of the line number map acquired by SourceGate through examination of the module's line number symbols (whose format is set with the SourceGate command line switch '-L' or through the Source Code Interface Options window), the module's source code file name, and the current debug mode set for that module (Assembly, Mixed, or Source-only).

:=M [module] Mixed Mode Command

The ':=M' command sets the named module (or all modules if none is named) to Mixed Debug mode. Once a module has been enabled for mixed debug mode, that module's source file will be opened, and the source lines for that module will be inserted at appropriate points into the output of all commands which disassemble code (i.e. the 'L' (List) command, 'SS' (single-step) command, etc.).

When the ':=M' command is used, SourceGate must be able to locate and successfully open each named module's source code file. These source code files are assumed to reside in the current directory, and their names are assumed to be the module's name, with the extent of ".C".

If any of these assumptions are false, SourceGate will prompt for a source code file name, giving the name of the module for which it cannot find the source code file. The full path name of the file, along with the corresponding drive designator, should be entered at this time.

If no source code file is available, entering no file name at the prompt will cause SourceGate to set that module back to Assembly mode debug (which requires no source code file), and continue on to the next module to be set to Mixed debug mode.

:=S [module] Source Mode Command

The ':=S' command sets the named module (or all modules if none is named) to Source-only Debug mode. When source-only debug mode is selected for a particular module, all assembly language displays for that module will be suppressed. (Some assembly may be displayed in the source only mode. This assembly is from the library routines of the compiler and is source code to the emulator since most of these routines are written in assembly language). In addition, that module's source file will be opened and the source lines for that module will be inserted into the output of all commands which disassemble code (i.e. the 'L' (list) command, 'SS' (Single-Step) command, etc.).

When the ':=S' command is used, SourceGate must be able to locate and successfully open each named module's source code file. These source code files are assumed to reside in the current directory, and their names are assumed to be the module's name, with the extent of ".C".

If any of these assumptions are false, SourceGate will prompt for a source code file name, giving the name of the module for which it cannot find the source code file. The full path name of the file, along with the corresponding drive designator, should be entered at this time.

If no source code file is available, entering no file name at the prompt will cause SourceGate to set that module back to Assembly mode debug (which requires no source code file), and continue on to the next module to be set to Source debug mode.

Available CPU32 Commands

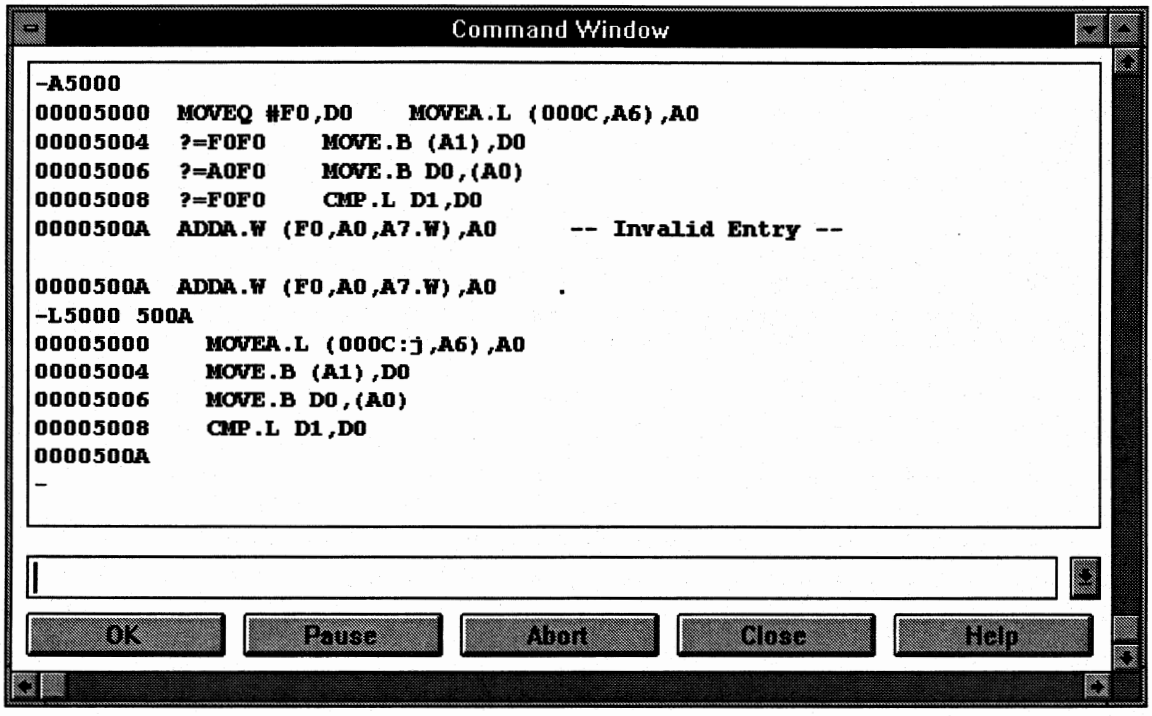
The following commands are available from the command line for this emulator:

A (In-Line Assembly)
BM (Byte Mode)
C (Compare Memory Block)
CM (Show Status Of Control Board Memory)
CONFIG (Recall Factory Configuration)
CONFIGT (Recall Factory Configuration For Target)
CRL (Load Configuration Registers)
D (Dump Memory Block)
DM (Data Memory Mode)
DW (Write Line Active For Target Writes Only)
E/EN (Enter/Change Memory)
EDT (Events Disabled For Temp. Breakpoints)
EET (Events Enabled For Temp. Breakpoints)
ET (Show Status For EDT/EET)
EW (Write Line Active For All Memory Writes)
EXP (Examine/Change Peripheral)
EXR (Examine/Change Peripheral Register)
F (Fill Memory Block)
G (Begin Real-Time Emulation)
HLD (Hex Listing Disabled)
HLE (Hex Listing Enabled)
I/IN (Input)
L (List Disassembled Code)
LM (Long Word Display Mode)
M (Move Memory Block)
MT/MTL (Test Block Of Memory)
O/ON (Output)
PD (Program Wait States For DSACK)
PM (Program Memory Mode)
RD/RDP/RDX (Read Object File Into Memory)
RP (Reset Processor, Continue Emulation)
RS (Reset Processor, Stop Emulation)
SH (Show Status For SHD/SHE)
SHD (Serial Handshake Disabled)
SHE (Serial Handshake Enabled)
SM (Supervisor Memory Mode)
SP (Stop Emulation)
SR/SRN (Search Block Of Memory)
SS (Single Step)
ST (Show Emulator Status)

UM (User Memory Mode)
VASM (View Assembly)
VER (Display Firmware Version)
W/WX (Write Memory Contents To File)
WM (Word Display Mode)
X (Examine/Change Registers)

All of the preceding commands are described in detail in the sections that follow. Optional parameters are shown enclosed by parenthesis.

A (In-Line Assembly) Command



This command allows assembly language mnemonics to be entered into memory starting at the specified address (In-Line Assembly). Once the Assembly command is entered, the address will be displayed with the cursor to the right. At this point, CPU32 assembly language instructions can be entered. After each entry, the emulator displays the next address and prompts for the next instruction. An error in the entered instruction will be flagged by displaying "-- Invalid Entry--" and re-displaying the current address. A period (.) followed by a carriage return terminates the entry sequence and the command.

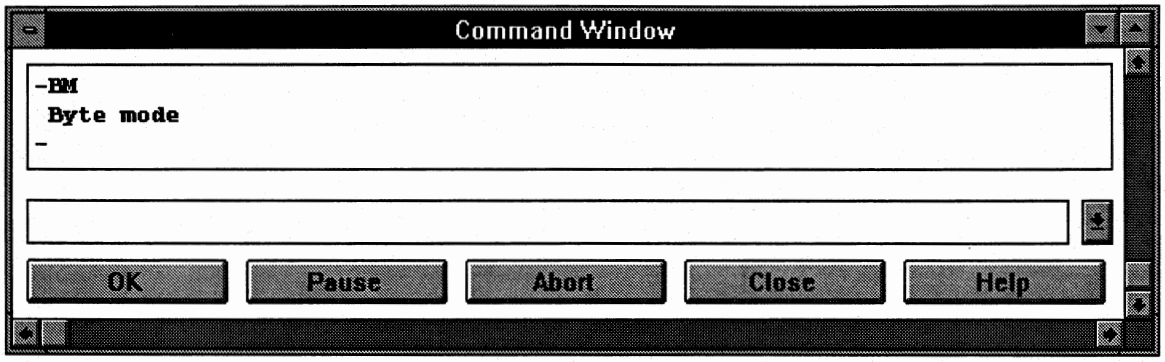
The format for the Assemble command is:

A([SUPDT]) Addr

where:

Addr:	is the hexadecimal starting address
SUPDT:	are the available overrides

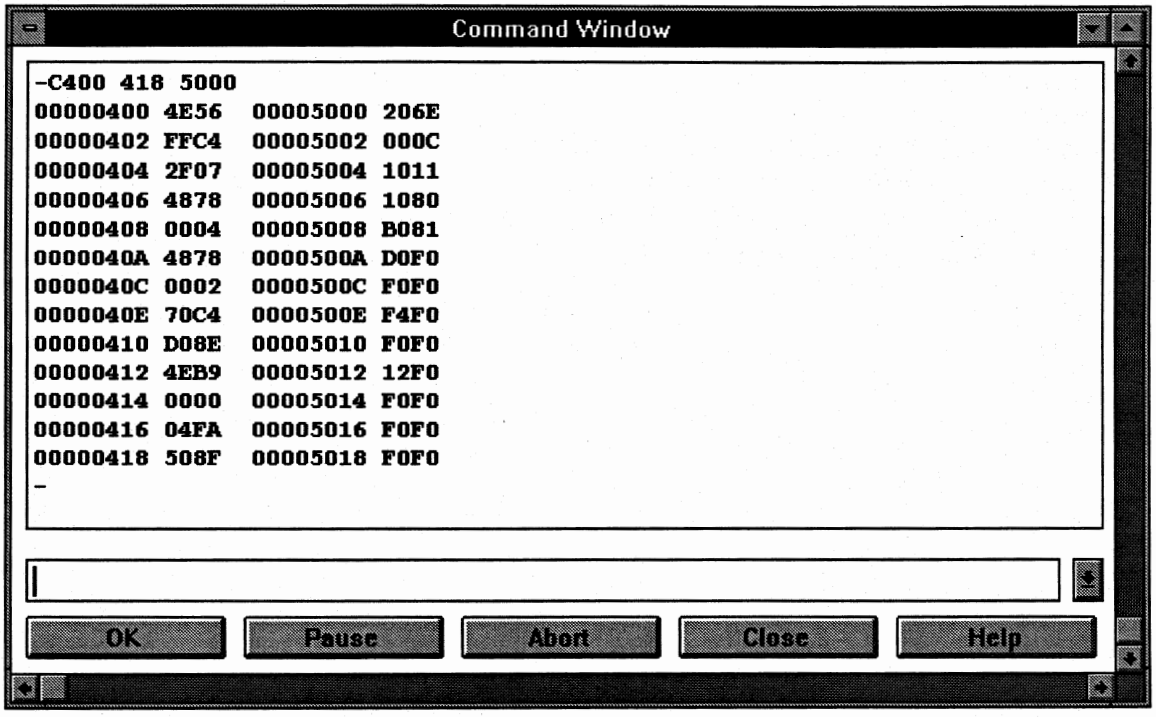
BM (Byte Mode) Command



This command sets the default for input or output values to "Byte Mode". While in Byte mode, commands such as D (Dump memory) will display values in byte mode instead of word mode or long word mode, which are the other choices (see WM and LM).

In addition, commands which require an input value will be affected by this command. For example, if the Fill command, F 8000 9000 33 is executed while in byte mode, each byte of memory from 8000H through 9000H will contain 33H. If this same command was executed with the emulator in word mode, then 0033H would be stored in each word location from 8000H to 9000H. If this same command was executed with the emulator in long word mode, then 00000033H would be stored in each long word location from 8000H to 9000H.

C (Compare Memory Block) Command



The Compare command will compare one block of memory to another. The first word in the first block will be compared to the first word in the second block, and so on. Any non-comparisons will be displayed on the screen showing the address in the first block with its data value and the address in the second block with its data value. If the two blocks match, then nothing will be displayed. **Note that you can also use the Memory window to compare memory.**

The format for the Compare command is:

C([SUPDTR]) Baddr1 Eaddr1 ([SUPDT])Baddr2

where:

Baddr1:	is the hexadecimal beginning address of the first compare block
Eaddr1:	is the hexadecimal ending address of the first compare block
Baddr2:	is the hexadecimal beginning address of the second compare block
SUPDTR:	are the available overrides

The size of the first block sets the number of words to be compared. For each word in block one that does not match the corresponding word in block two, a line is displayed on the screen.

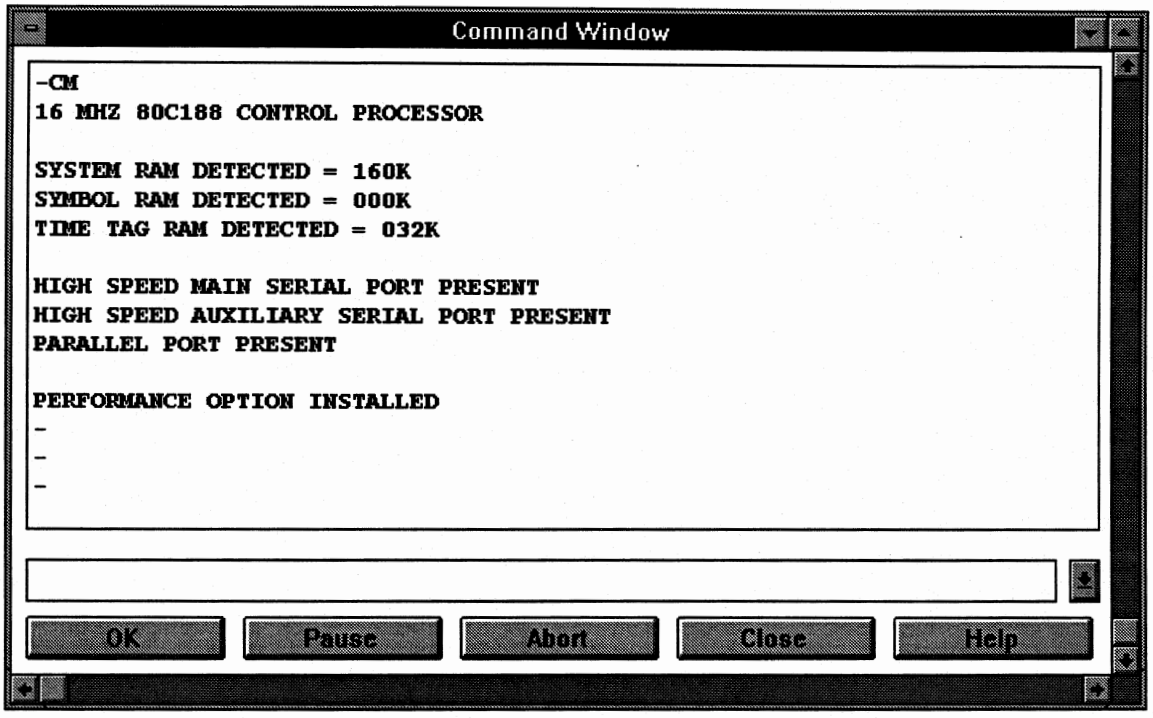
This display line has the format:

Addr1 Value1 Addr2 Value2

where:

Addr1:	is the address of the word in block one
Value1:	is the contents of addr1
Addr2:	is the address of the word in block two
Value2:	is the contents of addr2

CM (Show Status Of Control Board Memory) Command



Executing this command will show the emulator options that are installed. It should be noted that "System RAM" in this case refers to the system RAM inside the emulator (not the PC) and "Symbol RAM" refers to the optional 256K of RAM which allows symbolic debug when using the emulator with a dumb terminal. The SourceGate program does not use this RAM for symbolic or source-level debugging.

CONFIG (Recall Factory Configuration) Command

This command will return the emulator menus, ports, memory map, predefined sequences and other battery backed-up configuration values to the factory configuration.

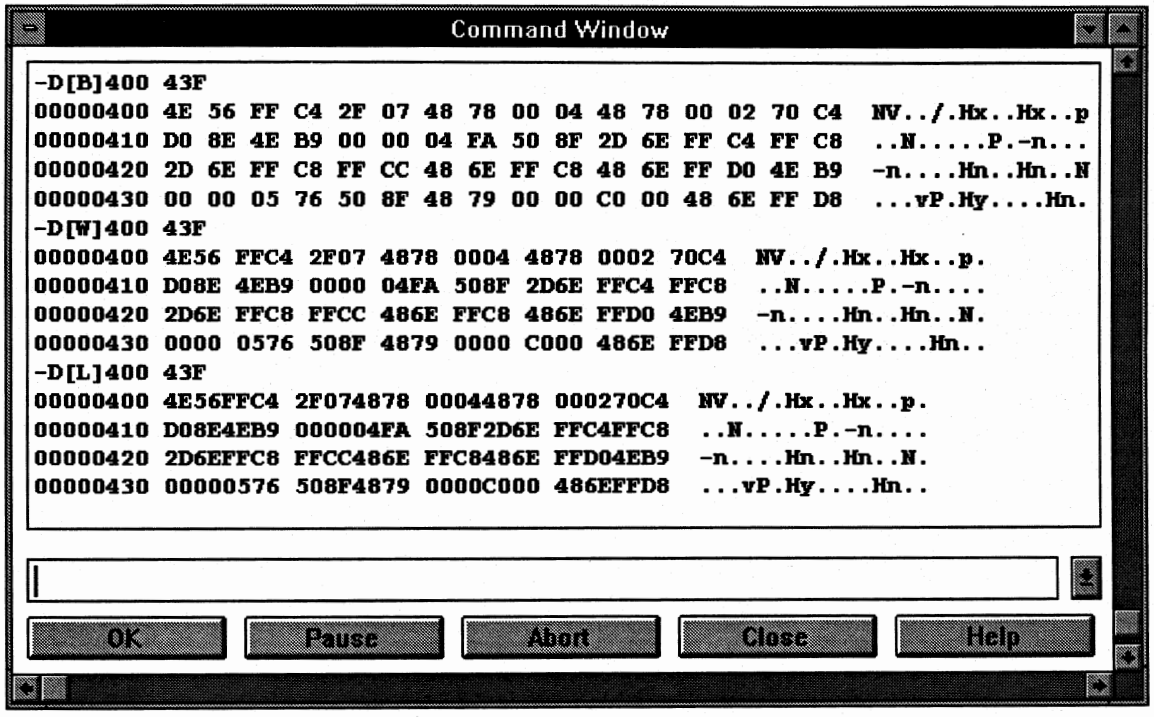
CONFIGT (Recall Factory Configuration For Target) Command

This command will return the emulator menus, memory map, predefined sequences, and other battery backed- up configuration values to the factory configuration for target system operation. When this command is executed, all parameters that can be set to come from the target system (clock source, memory, etc.) will be configured to come from the target system.

CRL (Load Configuration Registers) Command

This command will read the current values of the registers shown in the Configuration window and will save these values in battery-backed memory. Then, each time a CONFIG command is issued, these values will be re-read from memory and programmed into the processor and the Configuration window.

D (Dump Memory Block) Command



The Dump Command will display the contents of memory in the specified address range showing the hexadecimal data with its ASCII equivalent on the screen. If no ending address is given, then one screen of data will be displayed and the command will terminate. If no starting address is specified then the last ending address (whether user entered or default) will be used as the starting address. Using this method of defaulting allows you to dump a display of memory from one starting address and get the next and subsequent pages of data dumped by simply typing the "D" and a carriage return. **Note that you can also use the Memory window to display the contents of memory.**

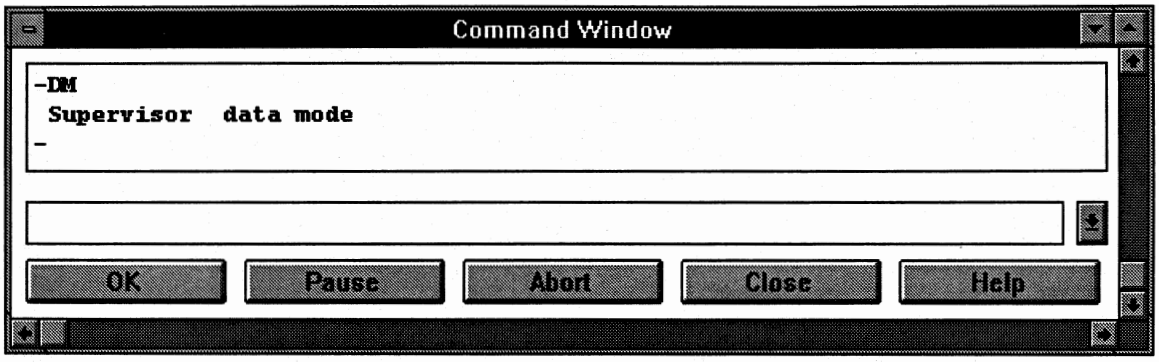
The Format for the Dump Command is:

D([SUPDLWBTR]) (Baddr) (Eaddr)

where:

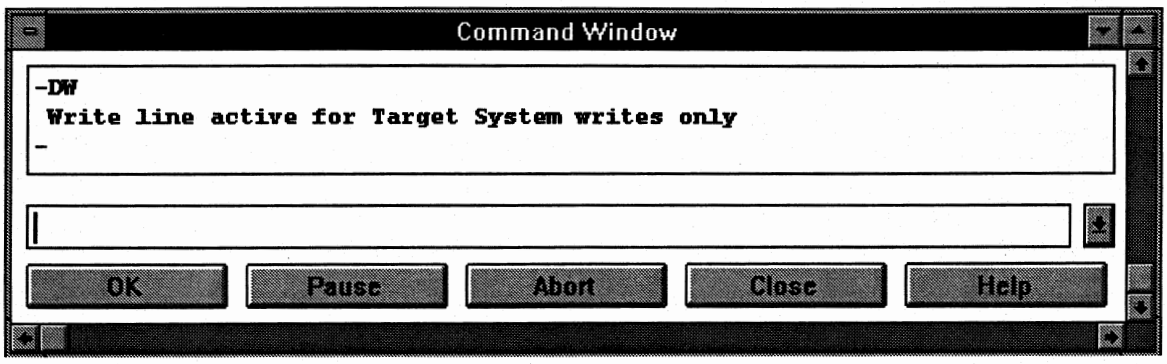
Baddr:	is the hexadecimal beginning address of the block of memory to be displayed
Eaddr:	is the hexadecimal ending address of the block of memory to be displayed
SUPDLWBTR:	are the available overrides

DM (Data Memory Mode) Command



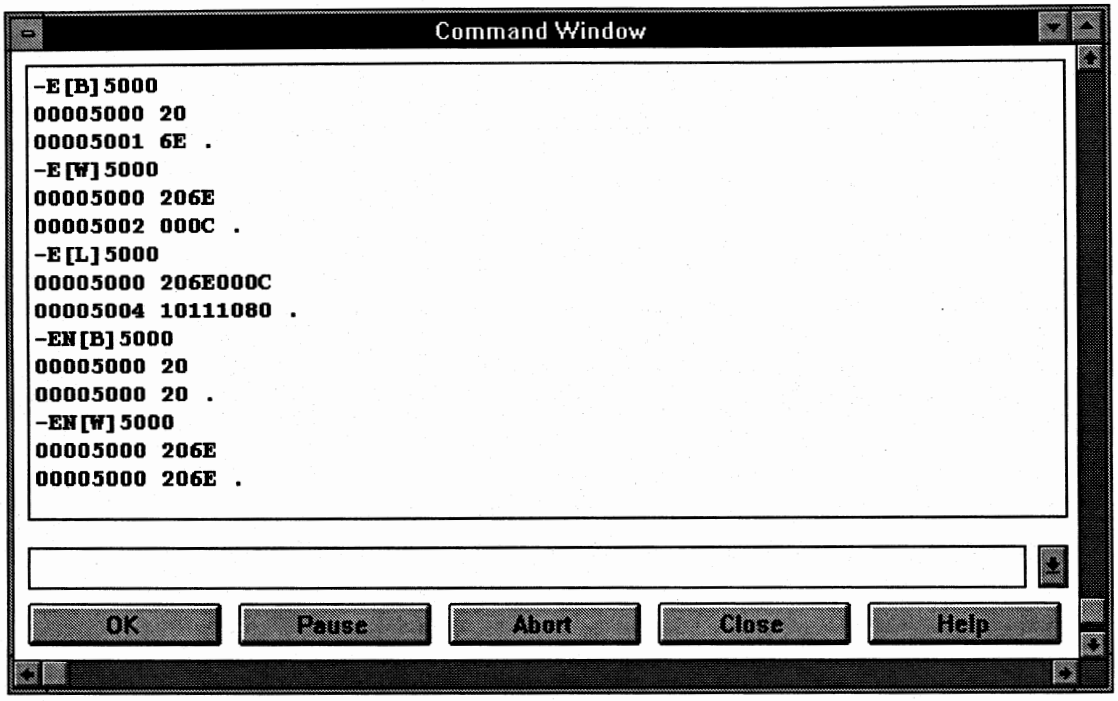
The execution of this command sets the default memory category to be Data Memory mode. Using the SM (Supervisor Mode) or UM (User Mode) commands further define the default memory mode.

DW (Write Line Active For Target Writes Only) Command



In normal instances, executing a command like "F" (Fill) on emulator memory will also fill the corresponding target system memory with the same data. The DW command will cause the write line to remain high at all times except during write cycles to addresses which are mapped as target system memory.

E/EN (Enter/Change Memory) Command



The Enter Command is used to examine memory at the specified location with the option to change its contents. After executing the Enter Command, the selected address will appear on the screen with the current data. The data can be altered by entering new data and pressing a carriage return, or it can remain unchanged by just typing a carriage return. In either case the next address will be displayed with its data. This command can be terminated by typing a period (.) in the data field followed by a carriage return. **Note that you can also use the Memory window to examine and change the contents of memory.**

There is a variation of the Enter command (EN) that does not advance to the next address. Whether you enter a change and a carriage return or just a carriage return, it will display the value at the specified address only. This feature is very useful in monitoring a memory mapped I/O location.

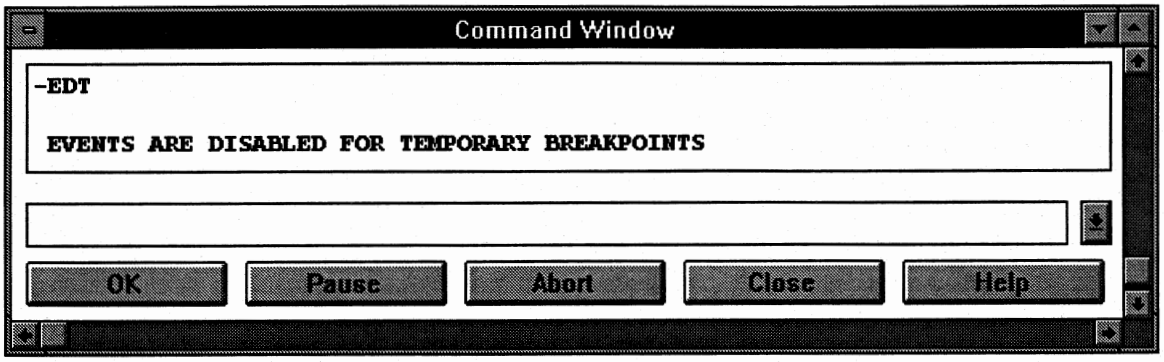
The Format for the Enter Command is:

E([SUPDLWBT]) Addr
or
EN([SUPDLWBT]) Addr

where:

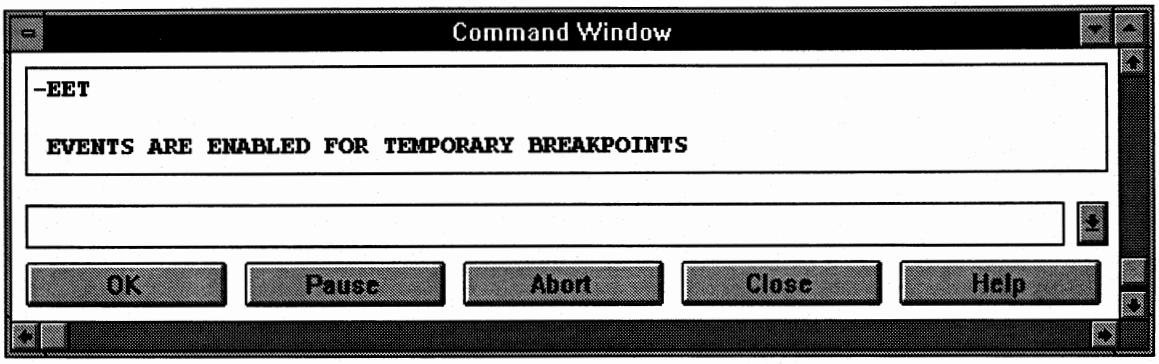
Addr:	is the hexadecimal address of the data value to be displayed for possible substitution
SUPDLWBT:	are the available overrides

EDT (Events Disabled For Temp. Breakpoints) Command



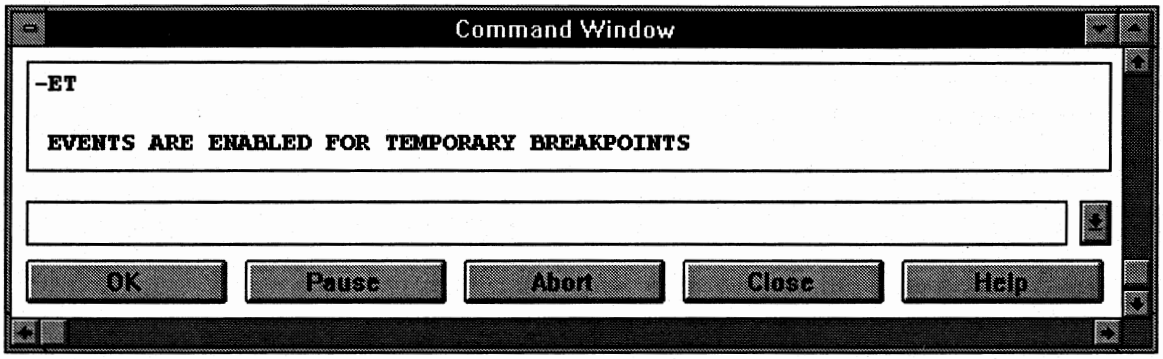
This command is for use with the Performance Analysis option only. The EDT command will override the breakpoints set in the event windows temporarily and use the breakpoints that are defined on the command line.

EET (Events Enabled For Temp. Breakpoints) Command



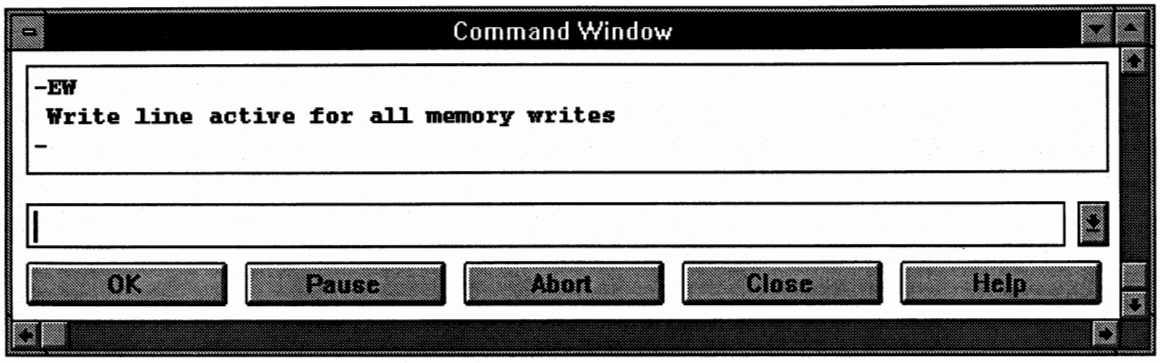
This command is for use with the Performance Analysis option only. The EET command will "OR" the breakpoints set in the event windows with the breakpoints that are defined on the command line.

ET (Show Status For EDT/EET) Command



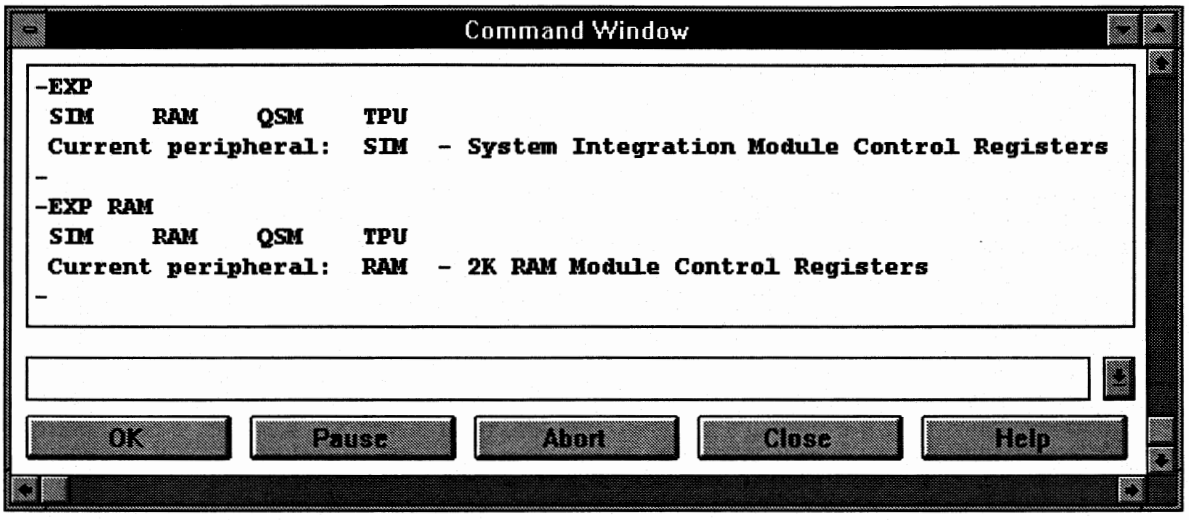
The ET command shows the current status for the EDT and EET commands.

EW (Write Line Active For All Memory Writes) Command



The EW command will allow the processor write signal to be enabled to the target system during any emulator command. For example, when the F (Fill) command is executed, the write signal will be passed through to the target system at the specified address regardless of the configuration of the memory map.

EXP (Examine/Change Peripheral) Command



The EXP command changes the current default peripheral. The EXR command can then be used to examine/change a specific register within the default peripheral. Entering EXP by itself on the command line will show the current default peripheral.

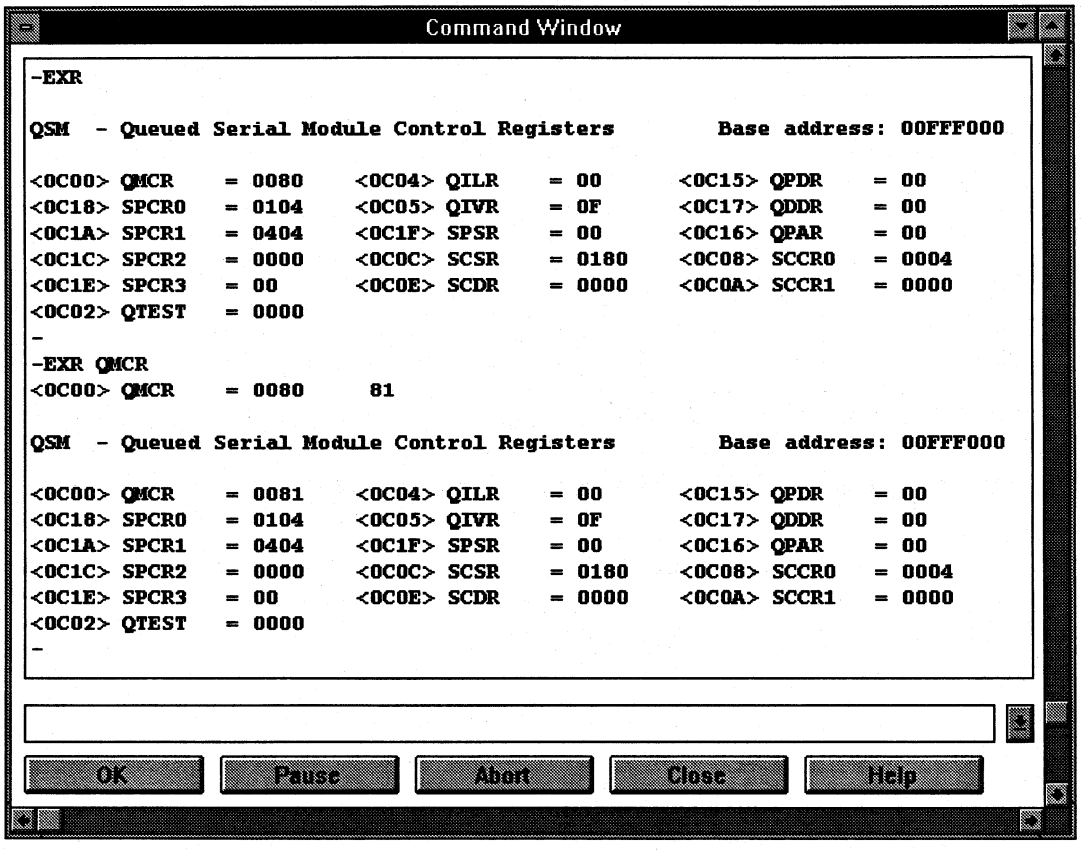
The format of the EXP command is:

EXP per

Where:

per is the desired default peripheral (QSM, RAM, SIM, or TPU)

EXR (Examine/Change Peripheral Register) Command



The EXR command allows a register in the current default peripheral to be examined/changed. The default peripheral can be selected by using the EXP command. Entering EXR by itself on the command line will show all registers associated with the current default peripheral.

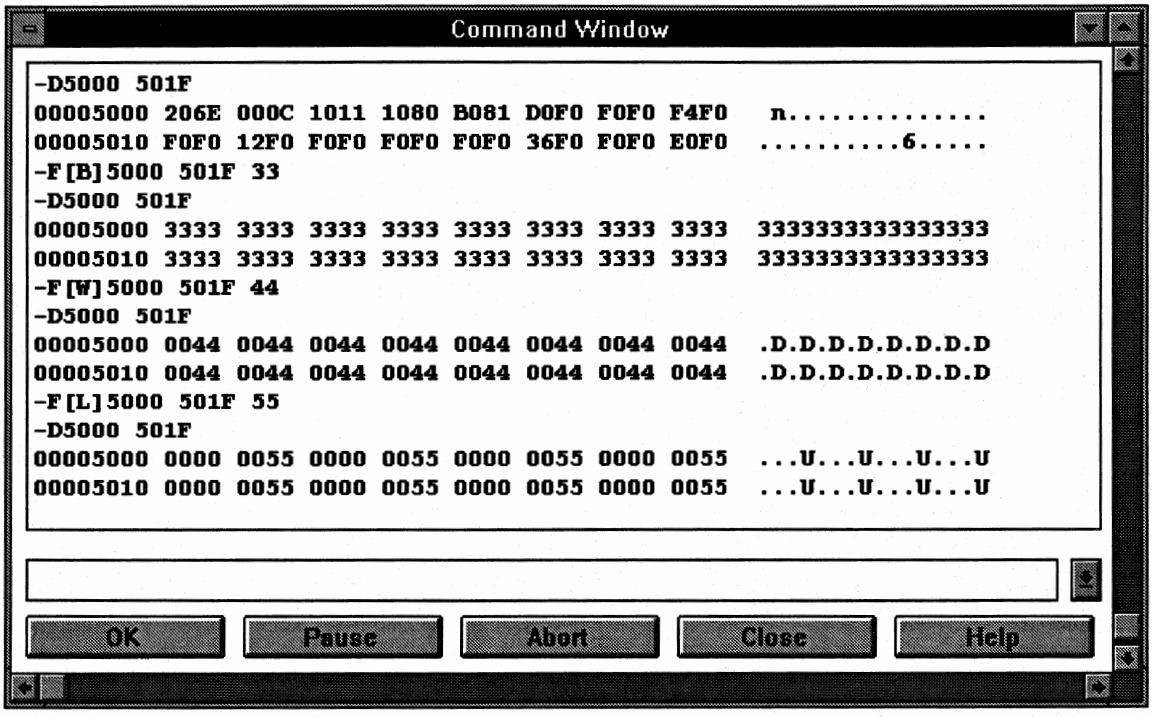
The format of the EXR command is:

EXR reg

Where:

reg is the register peripheral to be examined or changed

F (Fill Memory Block) Command



The Fill Command is used to fill memory with a constant. The given data value will be written to each memory location within the specified range. The value may be a byte or a word in length, determined by the global parameters and overrides used. **Note that you can also use the Memory window to fill memory.**

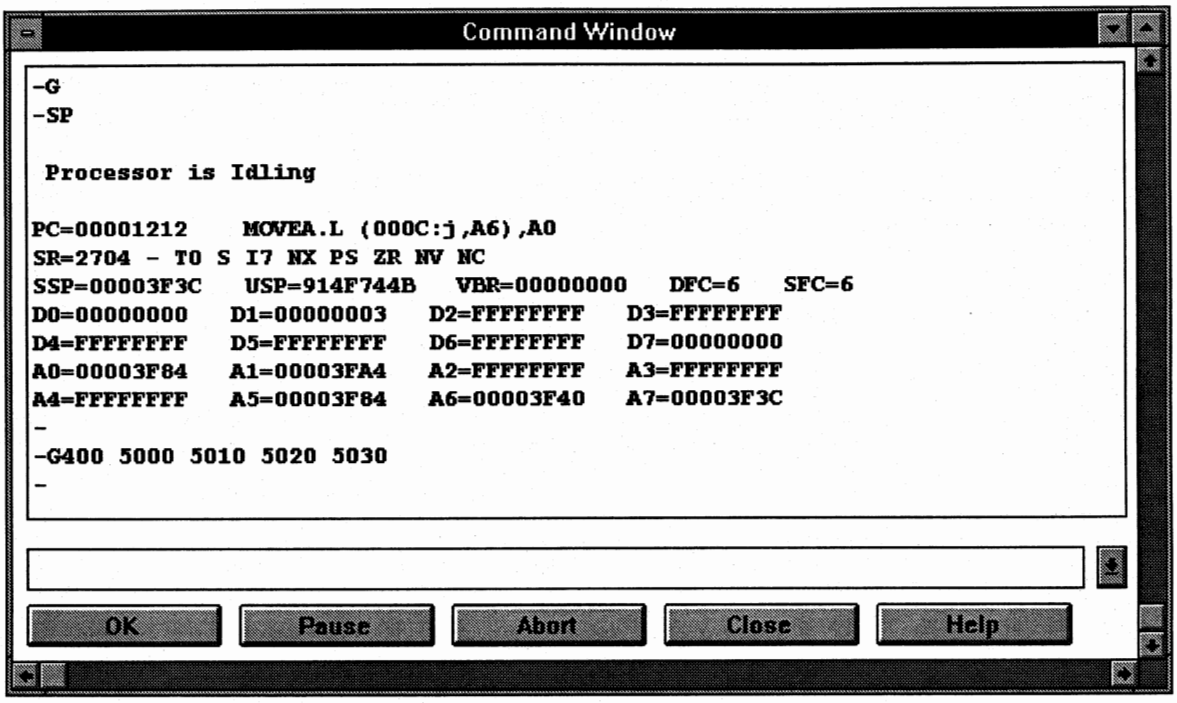
The format for the Fill Command is:

F([SUPDLWBTR]) Baddr Eaddr Val

Where:

Baddr:	is the hexadecimal beginning address of the memory block to be filled
Eaddr:	is the hexadecimal ending address of the memory block to be filled
Val:	Constant value to be used to fill the memory block
SUPDLWBTR:	are the available overrides

G (Begin Real-Time Emulation) Command



This command is used to start real-time emulation. The options for this command are to run with temporary breakpoints as set in this command, or without a breakpoint, in which case any break conditions which occur are due to the configuration of the events from the Event windows and the breakpoint sequence as defined in the Sequence Break Point window. If the option for running with breakpoints from this command is selected, then it is important to note that the configuration of the Event and the Sequence windows become inactive during execution of this command. In either case emulation will begin at the start address given or default to the current program counter. If a starting address is to be used it must be immediately adjacent to the G (otherwise it will be taken as a breakpoint address). **Note that you can also use the 'Source/Run' menu item along with the "Run Status" window to begin emulation.**

The format for utilizing the Event and Sequence menu configurations is:

G(Saddr)

The format for setting temporary breakpoints is:

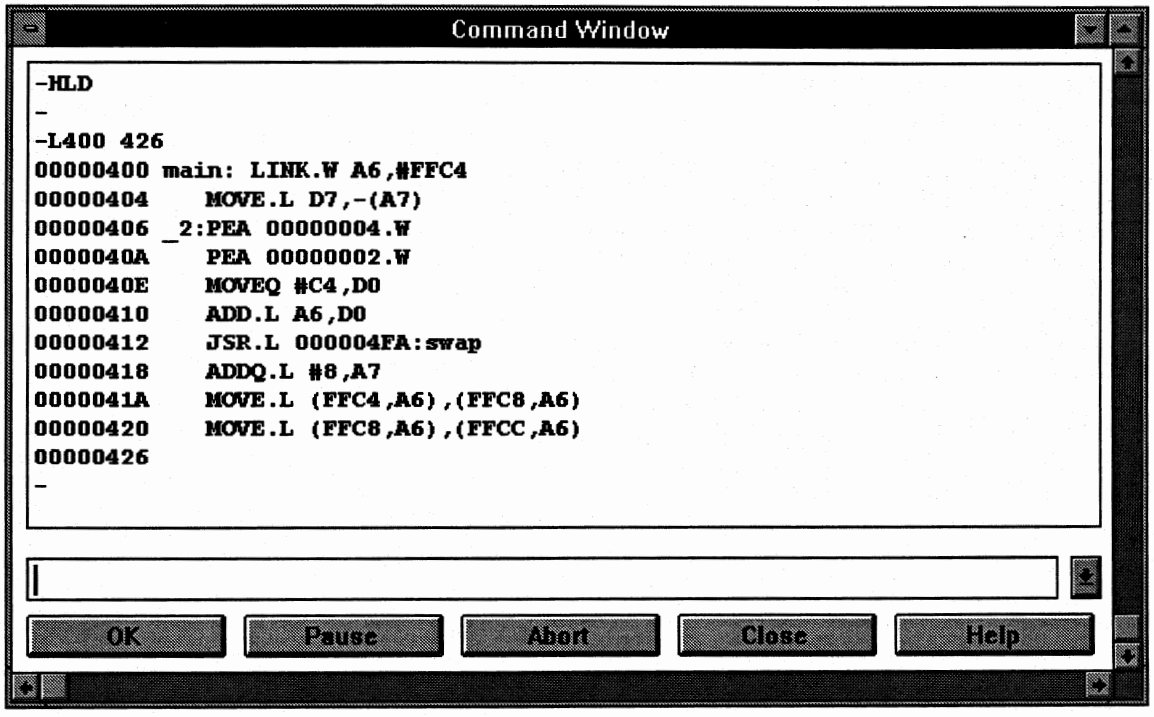
G(Saddr) Braddr1 (Braddr2 Braddr3 Braddr4)

Where:

Saddr: is the hexadecimal address where execution (real-time emulation) will begin

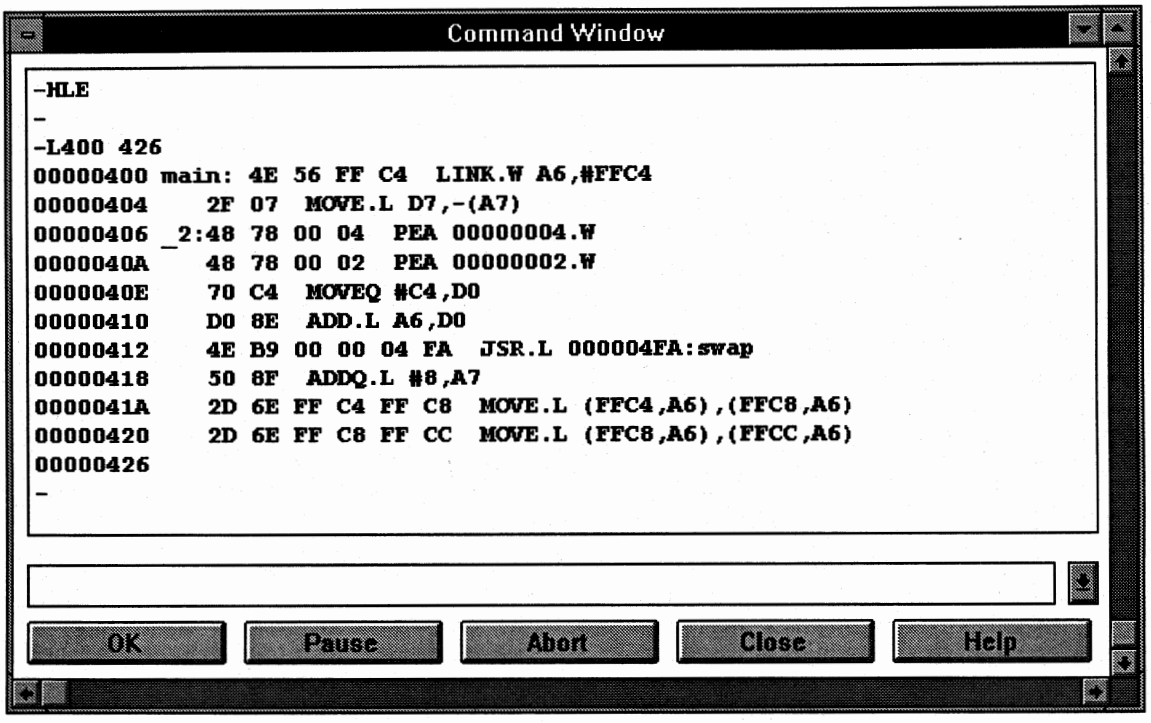
Braddr1-4: are hexadecimal addresses where execution (real-time emulation) will stop (or break) on op-code execution

HLD (Hex Listing Disabled) Command



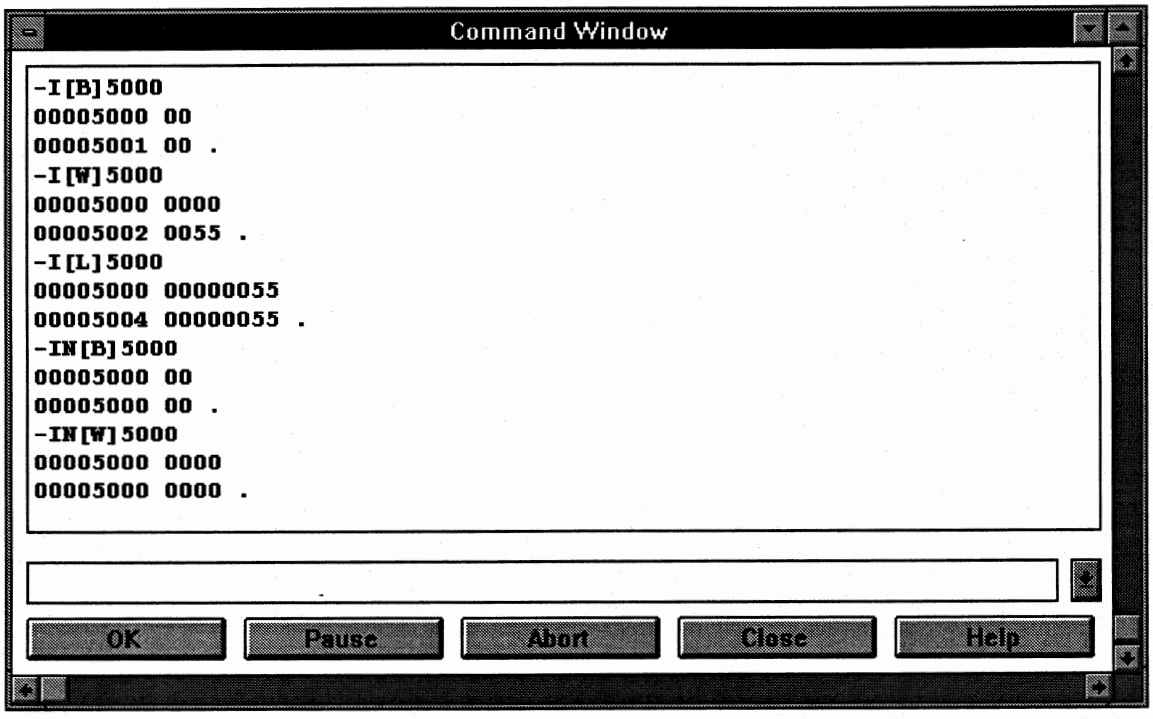
Executing this command will disable the hex information from being displayed with the disassembly information during the L (List) command. To enable the hex information, execute the HLE (Hex Listing Enabled) command.

HLE (Hex Listing Enabled) Command



Executing this command will enable the hex information to be displayed with the disassembly information during the L (List) command. To disable the hex information, execute the HLD (Hex Listing Disabled) command.

I/IN (Input) Command



The I command is used to read and display a certain memory address. This function is very useful when monitoring memory mapped I/O locations. Each time a <cr> is entered, the memory location is read, shown, and incremented to point at the next location. If you wish to suppress the incrementing of the address every time a <cr> is pressed, the IN command should be used. **Note that you can also use the Memory window to read and display memory.**

The format for the Input command is:

I([SUPDLWBT]) Addr

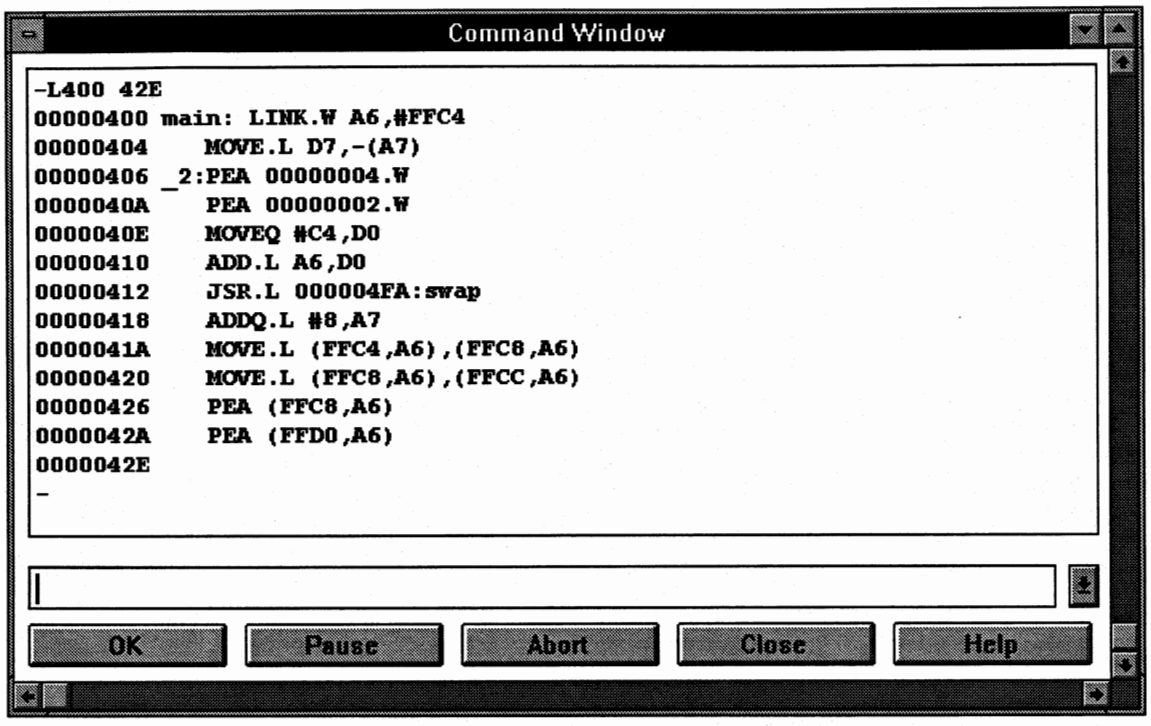
or

IN([SUPDLWBT]) Addr

Where:

Addr: is the hexadecimal address of the data value to be read
SUPDLWBT: are the available overrides

L (List Disassembled Code) Command



This command is used to disassemble the program memory data into assembly language mnemonics. All data and addresses are displayed in hexadecimal. If no ending address is specified, then one screen of disassembled mnemonics will be displayed and the command will terminate. If no beginning address is specified then the last ending address (whether entered or default) will be used as the beginning address. This allows you to disassemble code one display screen at a time by entering the command and an initial starting address for the first display and for the next and each subsequent page only the "L" and a carriage return need be entered. If a range is given, disassembled mnemonics will be displayed from the starting address to the ending address, scrolling the screen if necessary. To pause a scrolling screen enter CNTRL-S and, to resume scrolling a CNTRL-Q. This can also be accomplished by using the "pause" key to stop scrolling and <cr> to resume scrolling. To terminate the command while scrolling depress the space bar or enter a carriage return.

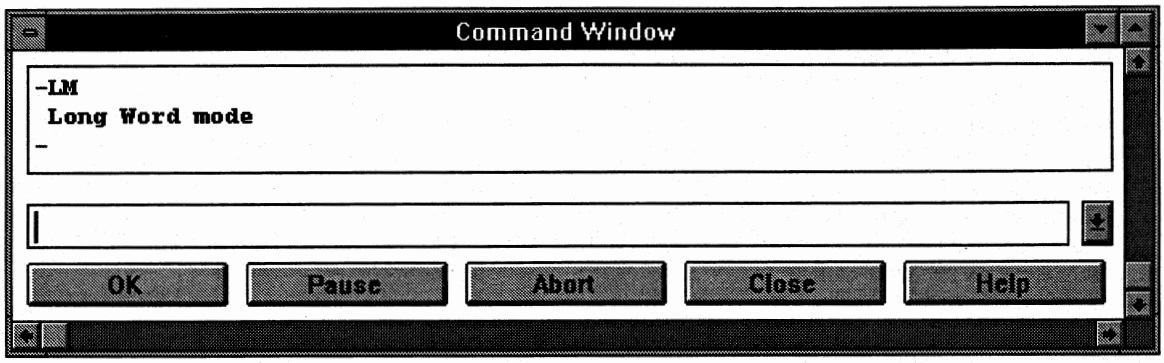
The format for the List Command is:

L([SUPDTR]) (Baddr) (Eaddr)

Where:

Baddr:	is the hexadecimal beginning address of the memory data to be disassembled
Eaddr:	is the hexadecimal ending address of the memory to be disassembled
SUPDTR:	are the available overrides

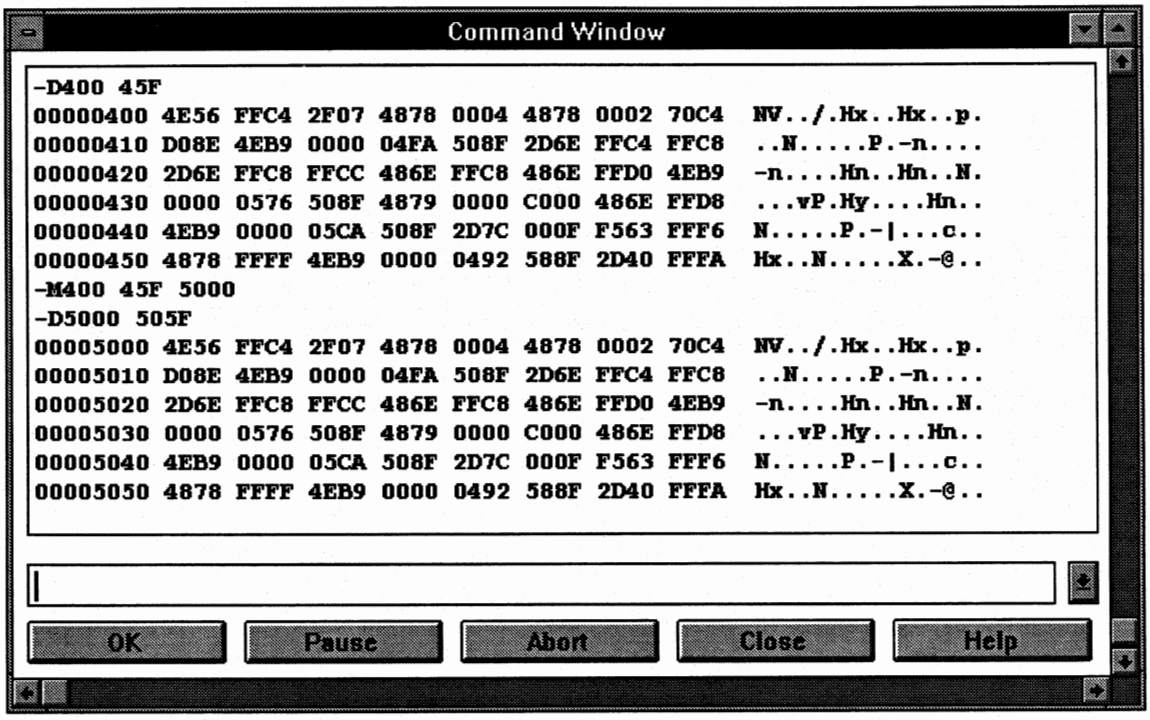
LM (Long Word Display Mode) Command



This command sets the default for input or output values to "Long Word Mode". While in Long Word mode, commands such as D (Dump memory) will display values in Long Word mode instead of word mode or byte mode, which are the other choices (see WM and BM).

In addition, commands which require an input value will be affected by this command. For example, if the Fill command, F 8000 9000 33 is executed while in Long Word mode, each long word of memory from 8000H through 9000H will contain 00000033H. If this same command was executed with the emulator in word mode, then 0033H would be stored in each word location from 8000H to 9000H. If this same command was executed with the emulator in byte mode, then 33H would be stored in each byte location from 8000H to 9000H.

M (Move Memory Block) Command



The Move Command is used to copy the contents of one block to the contents of another block (non- destructive move). Block one is defined with its starting and ending addresses being the first two address fields displayed, and block two is defined with its starting address being the third address field. The contents of the first memory location of block one is copied to the first memory location of block two, and so on. The next prompt character is not displayed until the move is complete. **Note that you can also use the Memory window to move memory.**

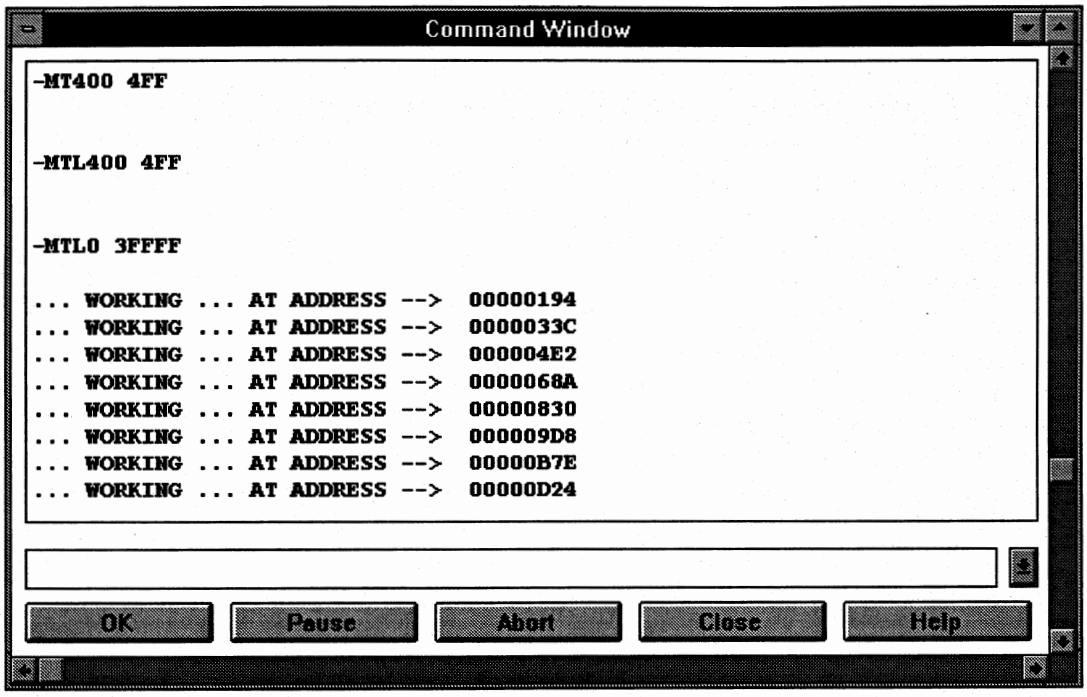
The format of the Move Command is:

M([SUPDTR]) Baddr1 Eaddr1 ([SUPDT])Baddr2

where:

Baddr1:	is the hexadecimal beginning address of the data "Source" block
Eaddr1:	is the hexadecimal ending address of the data "Source" block
Baddr2:	is the hexadecimal beginning address of the data "Destination" block . The size of the "Source" block defines the number of bytes to be copied into the "Destination" block.
SUPDTR:	are the available overrides

MT/MTL (Test Block Of Memory) Command


Command Window
-MT400 4FF
-MTL400 4FF
-MTL0 3FFFF
... WORKING ... AT ADDRESS --> 00000194
... WORKING ... AT ADDRESS --> 0000033C
... WORKING ... AT ADDRESS --> 000004E2
... WORKING ... AT ADDRESS --> 0000068A
... WORKING ... AT ADDRESS --> 00000830
... WORKING ... AT ADDRESS --> 000009D8
... WORKING ... AT ADDRESS --> 00000B7E
... WORKING ... AT ADDRESS --> 00000D24
OK
Pause
Abort
Close
Help

This command can be used to test memory (RAM) within the given address range. The short memory test (MT) writes, reads and compares two separate bit patterns for each byte within the range. The bit patterns are 55H and AAH, thus cycling all bits of the byte through both one and zero. The long memory test (MTL) is a walking bit pattern. This test writes, reads and compares the byte while having only one bit in the byte on at a time. This is done by using the Hexadecimal values 1, 2, 4, 8, 10, 20, 40, 80 for each byte within the range. In either test, any read not comparing with what was written into a location will cause the address, data written and data read to be displayed on the next line of the screen. Both of these tests are non- destructive and can be performed without corrupting data currently in memory. A <cr> will terminate this test at any time. **Note that you can also use the Memory window to test memory.**

The format for the Memory Test command is:

MT([SUPDTR]) Baddr Eaddr

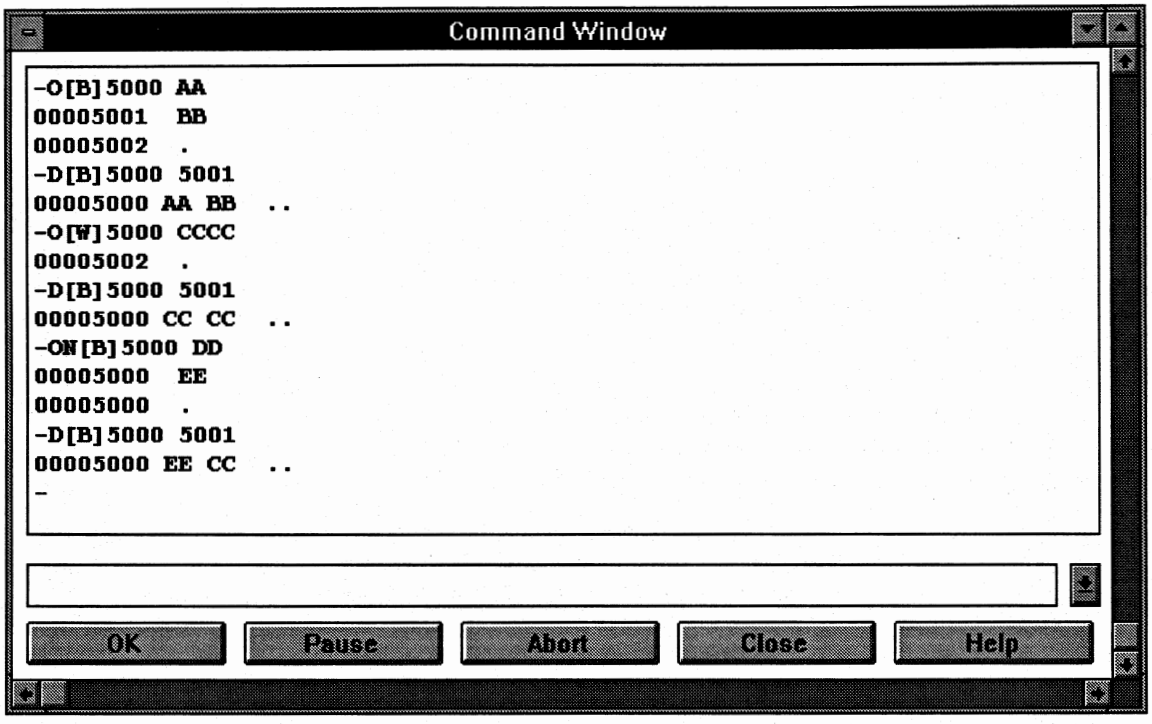
The format for the long Memory Test command is:

MTL([SUPDTR]) Baddr Eaddr

Where:

Baddr:	is the hexadecimal beginning address of the memory block to be tested.
Eaddr:	is the hexadecimal ending address of the memory block to be tested.
SUPDTR:	are the available overrides

O/ON (Output) Command



The O command is used to write to a certain memory address. This function is very useful when testing memory mapped I/O locations. Each time a <cr> is entered, the desired value is written to the displayed address and the address is incremented. The ON command will not increment the address to be written but will write the same address each time a <cr> is entered. **Note that you can also use the Memory window to output the contents of memory.**

The format for the Output command is:

O([SUPDLWBT]) Addr,Val

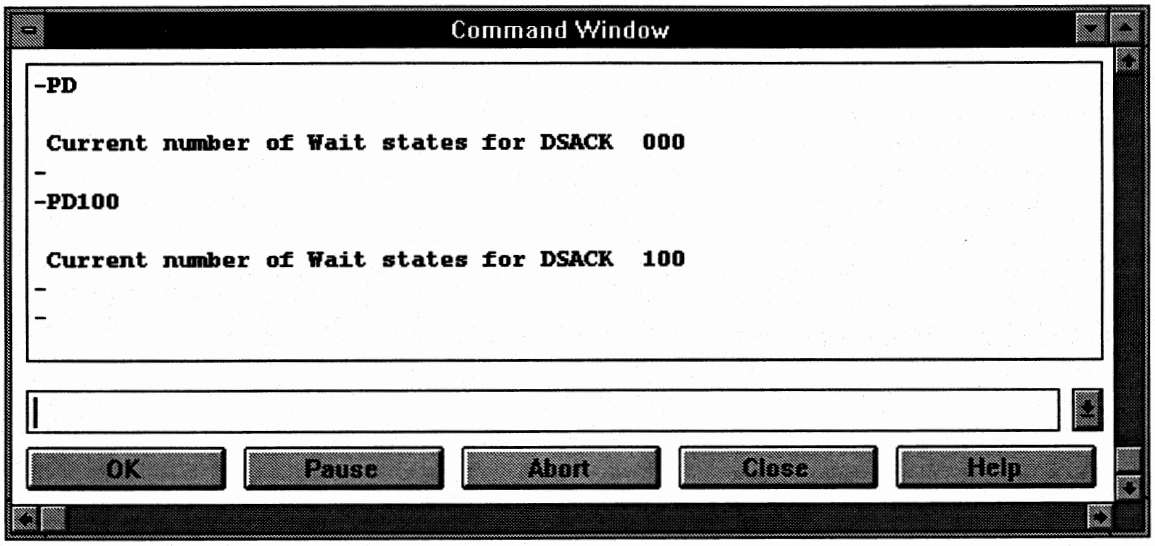
or

ON([SUPDLWBT]) Addr,Val

Where:

Addr:	is the hexadecimal address to be written with the defined value
Val:	is the hexadecimal value to be written
SUPDLWBT:	are the available overrides

PD (Program Wait States For DSACK) Command



This command is used to program the emulator to insert a predefined number of wait states into each memory cycle. Each wait state is one clock cycle in length. The wait states are inserted on any memory cycle in which the emulator is supplying the DSACK signal. The wait states are programmed by typing:

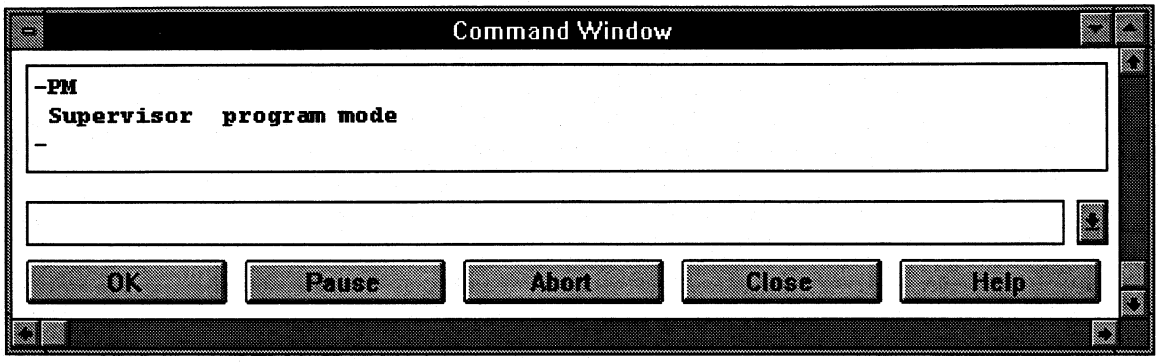
PD count

Where:

count: is the number of wait states to be inserted within the range of 0-254

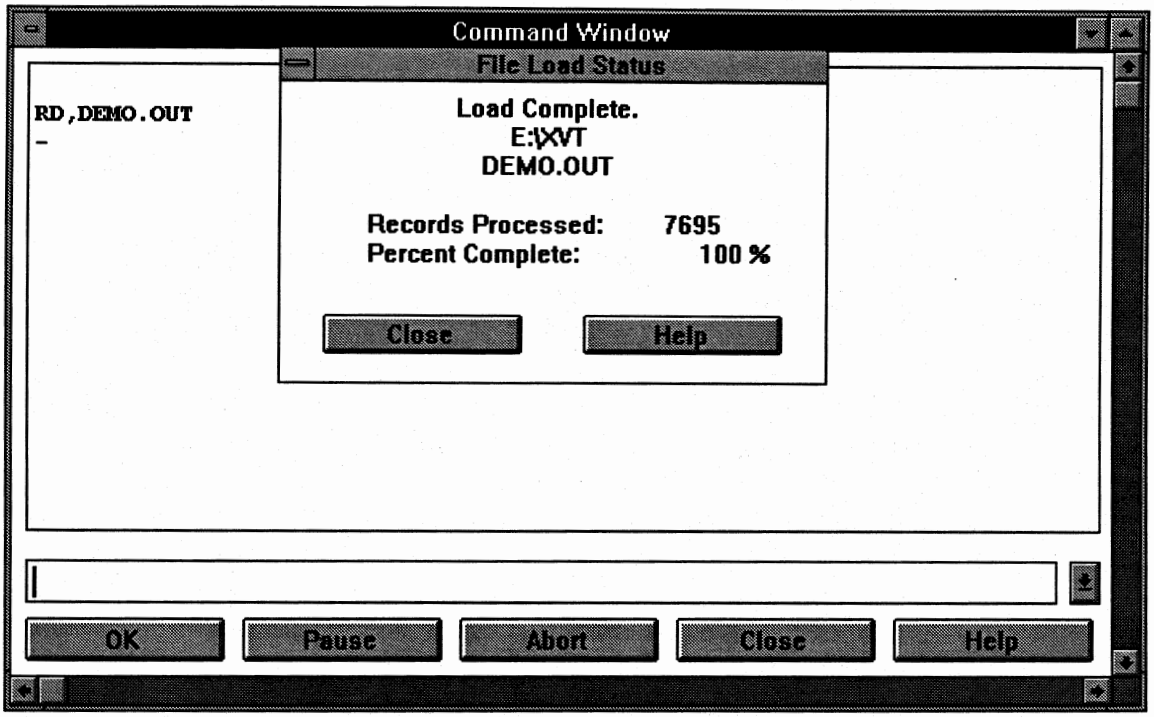
Typing PD<cr> will show the current number of wait states selected.

PM (Program Memory Mode) Command



The execution of this command sets the default memory category to be Program Memory mode. Using the SM (Supervisor Mode) or UM (User Mode) commands further define the default memory mode.

RD/RDP/RDX (Read Object File Into Memory) Command



This command is used to transfer a hexadecimal file from a host computer to the emulation memory. The file name can be specified with this command. When downloading a file using SourceGate II, the ASCII file is converted to binary before being transferred to the emulator. This essentially doubles the speed of the transfer over the normal ASCII transfer. When downloading from a host which doesn't use SourceGate, a standard ASCII transfer will occur. **Note that you can also use the 'File/Download Hex File' menu item to read a file.**

The file formats accepted by the SourceGate software are Motorola S-Record, Intel Hex, Texhex, HMI Binary, Extended Tek Hex, Microtec Research Absolute, Common Object File Format (COFF), Archimedes Debug (IAR), and Hitachi.

Once the Read Command has been executed, a File Load Status window will appear which will show the status of the read operation. The read operation can be aborted by pressing the **Cancel** button located in the File Load Status window.

When reading a file from the main RS232 port, the format is:

RD([SUPDT])(Offset),Filename,(Flags)

When reading a file from the auxillary RS232 port, the format is:

RDX([SUPDT])(Offset),Filename,(Flags)

When reading a file from the parallel port, the format is:

RDP([SUPDT])(Offset),Filename,(Flags)

Where:

SUPDT: are the available overrides

Offset: is a hexadecimal value representing the address offset

Filename: is a file name, valid to the host computer, that designates the file to be downloaded to the emulator. If no extension is specified, the default .ABS will be used.

Flags: are options to be used for each read. The options are defined as follows:

- A: Download hex files to the emulator using ASCII file transfer
- B: Download hex files to the emulator using binary file transfer
- C: Concatenate any symbols found reading a hex file with those which already exist in the internal symbol table (i.e. do not clear the symbol table on a read command)
- D: Use only the digits presented when looking up a symbol by value for display (Opposite of "Z")
- H: Upload/Download only the hex data records present in a hex file. Ignore all symbol records.
- S: Upload/Download only the symbol records of a hex file. Ignore all hex data records.
- M: Toggle between reading "S" record files as Motorola and Hitachi formats. Default is Motorola for Motorola microprocessors.
- Z: Pad a number with leading zeros when doing a lookup by value for symbol display (Opposite of "D").

The default option control flag settings are "BZ".

The 'RD' command opens the named format file (.ABS being the default extension) and reads its symbols into the internal SourceGate symbol table, and its hex data into the emulator's (or target system's) memory.

If the optional read bias is given, this offset value will be added to the addresses of all bytes read from the file, before these bytes are stored in memory. This offset is a hexadecimal number. If the filename is not given, the 'RD' command will use the filename given in the last 'RD' command.

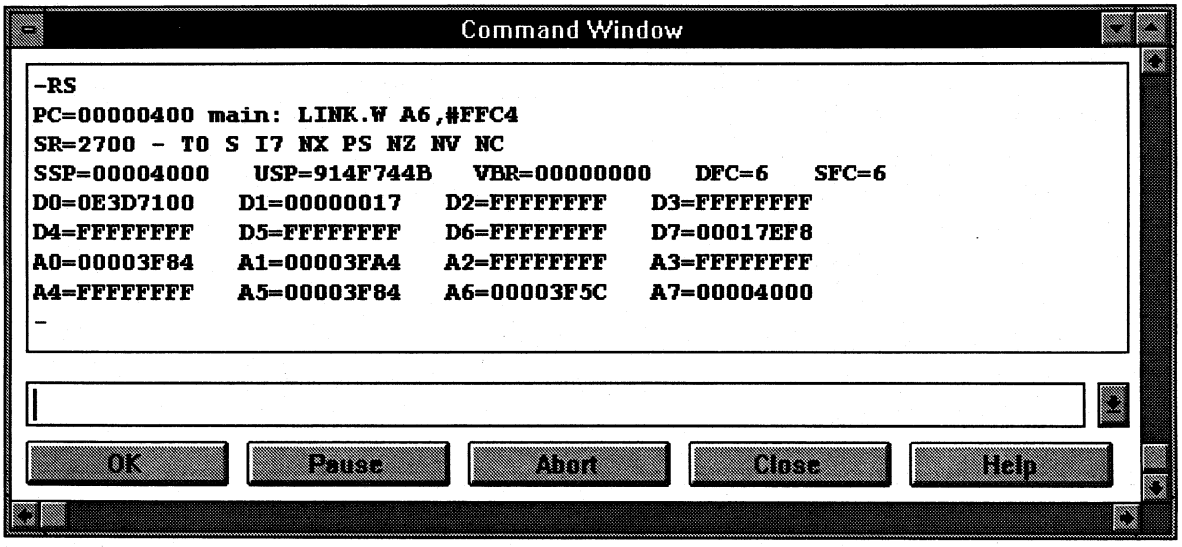
Note: The default file extension used on read (load) and write (dump) operations can be changed by modifying the .SG2 file for your emulator. For read operations, the "DefaultLoadExtension" definition can be changed to reflect the desired extension. For write operations, change "DefaultDumpExtension".

RP (Reset Processor, Continue Emulation) Command

The RP command allows you to reset the processor without stopping emulation.

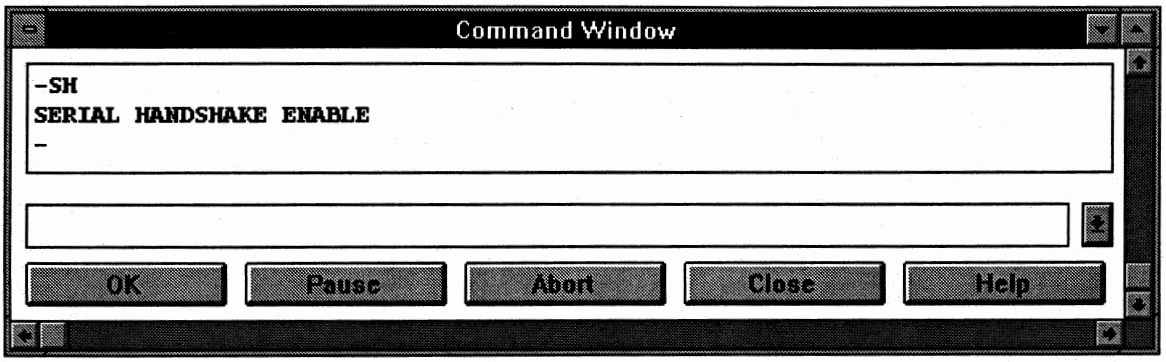
Note: The RP command can only be executed while emulation is in progress.

RS (Reset Processor, Stop Emulation) Command



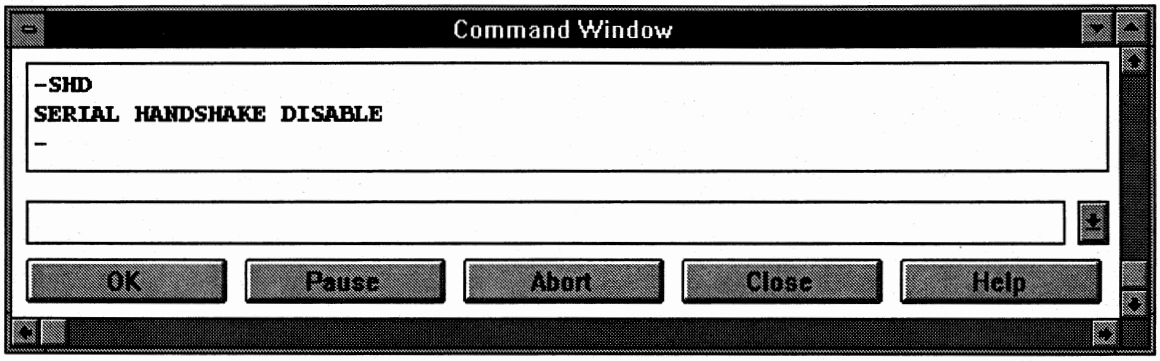
This command will issue a reset to the CPU32 processor. All registers will be updated to the value which they contain upon a processor reset. The Program Counter is initialized to the contents of memory locations 4H and 6H and the Stack Pointer is initialized to the contents of memory locations 0H and 2H. **Note that you can also use the 'Source/Reset Emulator' menu item and the "Run Status" window to reset the processor.**

SH (Show Status For SHD/SHE) Command



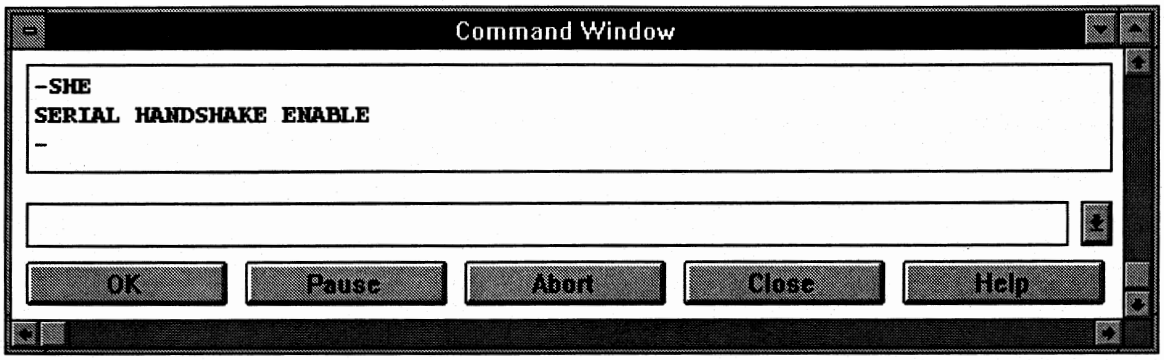
This command will show the status of the hardware handshake of the serial link between the emulator and the computer.

SHD (Serial Handshake Disabled) Command



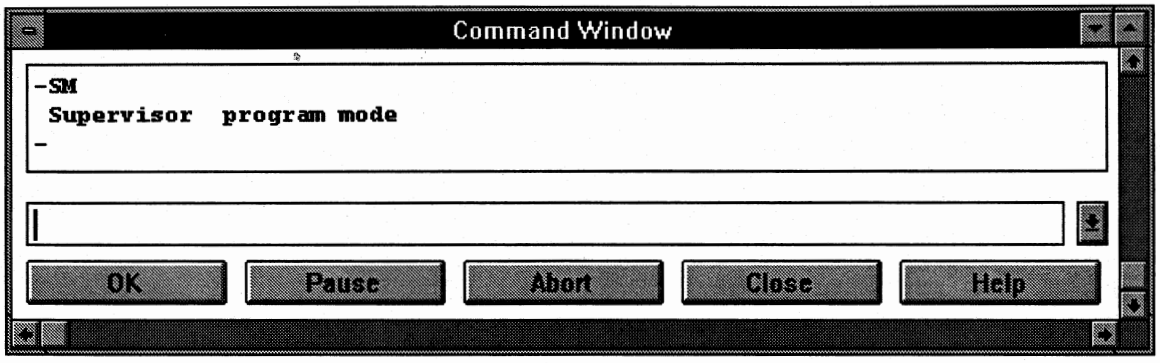
This command will disable the serial hardware handshake (RTS line) between the emulator and the computer. Using the SHE command or invoking SourceGate with the "-RTS" switch (which is the SourceGate default) will enable the hardware handshaking. This hardware handshaking may be needed to operate the emulator at the higher baud rates.

SHE (Serial Handshake Enabled) Command



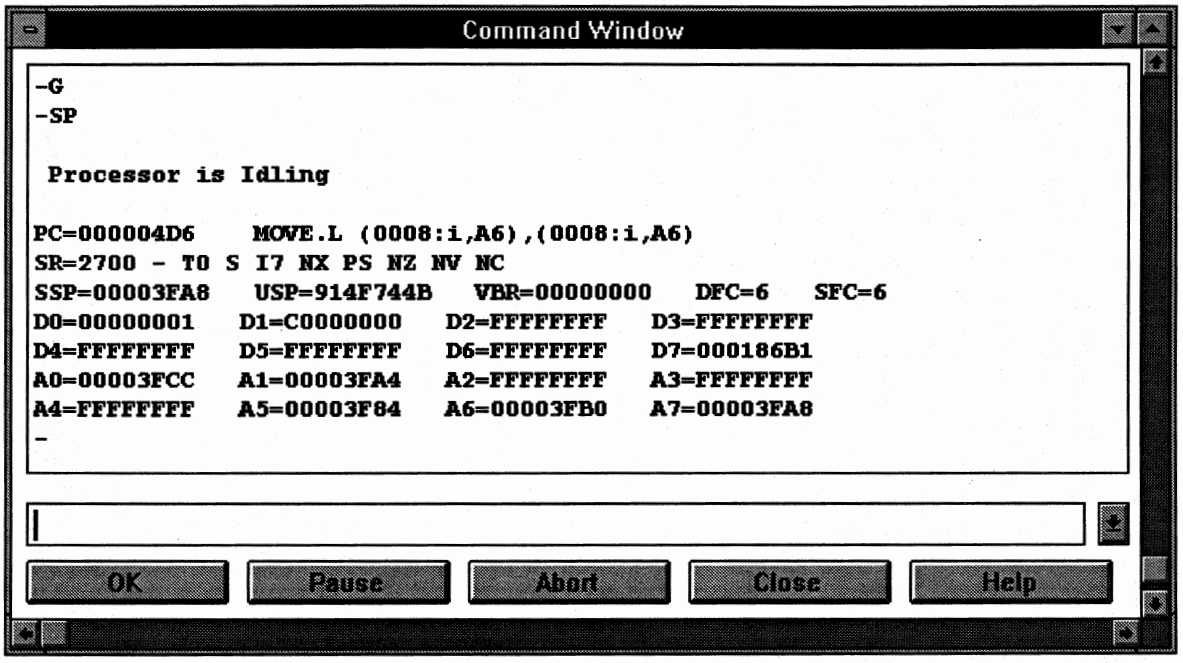
This command will enable the serial hardware handshake (RTS line) between the emulator and the computer. Using the SHD command or invoking SourceGate with the "-NORTS" switch will disable the hardware handshaking. This hardware handshaking may be needed to operate the emulator at the higher baud rates.

SM (Supervisor Memory Mode) Command



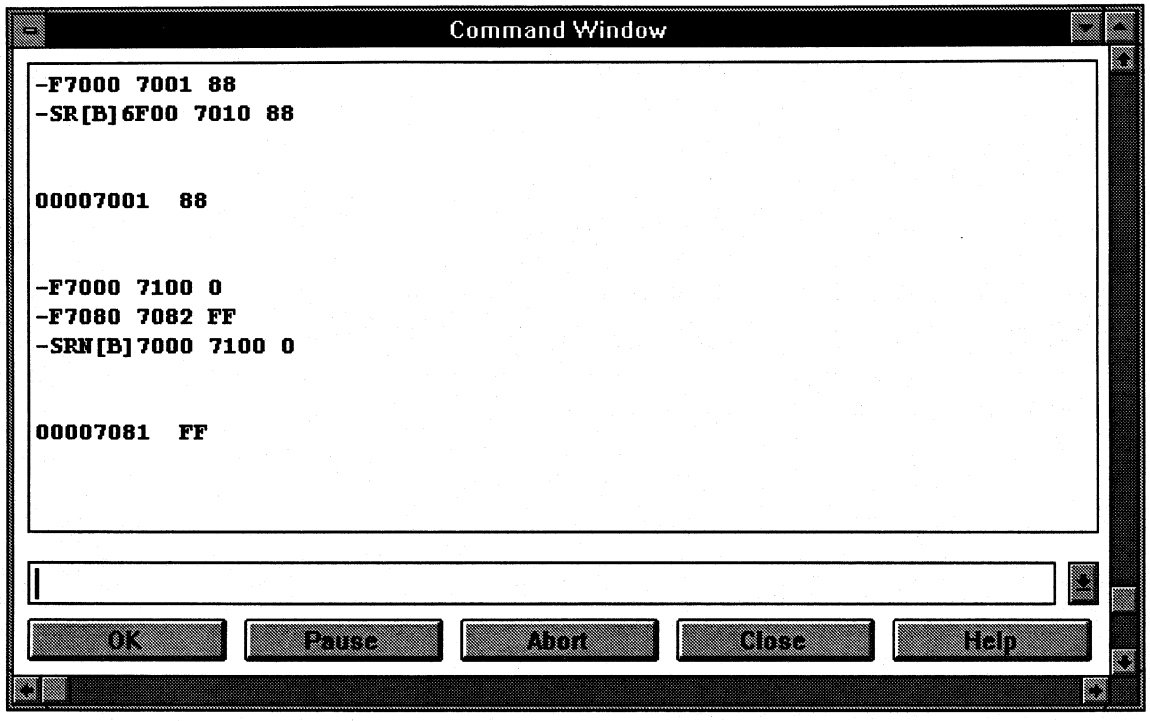
The execution of this command sets the default memory category to be Supervisor Memory mode. Using the PM (Program Mode) or DM (Data Mode) commands further define the default memory mode.

SP (Stop Emulation) Command



This command is used to stop emulation in order to regain control of the processor. Upon entering this command, emulation will break, and the message "PROCESSOR IS IDLING" will be displayed and the register window will be updated with the current register values. **Note that you can also use the 'Source/Stop' menu item and the "Run Status" window to stop emulation.**

SR/SRN (Search Block Of Memory) Command



This command can be used to search memory between two addresses for a particular data pattern. When a match is found, the address is displayed. To search for the next occurrence, hit the Carriage Return key. To terminate this command, type a period. If no match is found, then nothing is displayed. **Note that you can also use the Memory window to search the contents of memory.**

There is also a Search Not option available. In this mode the value is searched for, but the command displays the address of all memory locations in the given range that do not match the search value. This is useful when checking a block of memory that should contain a constant value, such as zero or the value of a previous Fill command.

The format for the Search command is:

SR([SUPDLWBTR]) Baddr Eaddr Val

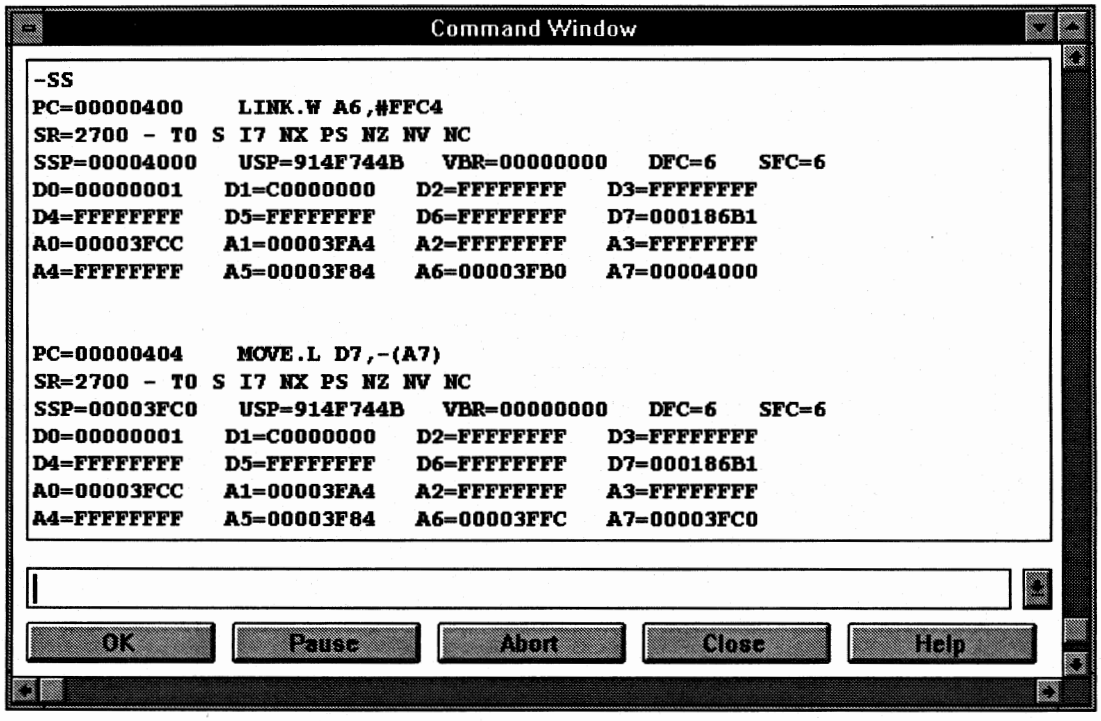
The format for the Search Not command is:

SRN([SUPDLWBTR]) Baddr Eaddr Sval

where:

Baddr:	is the hexadecimal beginning address of the block of memory to be searched
Eaddr:	is the hexadecimal ending address of the block of memory to be searched
Val:	is the data value to be searched for and will be 1 byte or word value or a 1-20 character ASCII string enclosed in single quotes.
Sval:	is the data value to be searched for and will be a byte value, a word value or one or two ASCII characters enclosed in quotes.
SUPDLWBTR:	are the available overrides

SS (Single Step) Command



This command is used to execute one instruction at a time. Each time a step is performed, the address and instruction to be executed are displayed along with the register values from the previous instruction executed. Therefore, multiple steps will display a history of program flow. There are four options for operating the Single Step Command. The first option is to single step every time a carriage return is entered. To terminate this mode, type a period. The second option is to single step a specified number of times. The third option is to single step continuously. The fourth option allows you to single step "Until" some condition. To terminate the second, third, and fourth options, a carriage return should be entered.

The format of the single step command is as follows:

SS[<count>][X] [Until clause] [Q/Q1/Q2] [Skip]

Where:

SS:	Steps once each time a <cr> is entered. A period (.) aborts stepping
SS<count>:	Steps <count> times without waiting for a <cr>. A <cr> aborts stepping
SSX:	Steps continuously until a <cr> is entered
SS Until clause:	Steps until the condition in the clause is true. A <cr> aborts stepping

The "Quiet" option is invoked by appending a "Q" to the end of the command. In this mode, only the first and last step are displayed (e.g. SSX Q will step automatically until a <cr> is entered showing the first and last instructions stepped). If "Q1" is used, only the last step is shown. If "Q2" is used, all steps are shown in the form "Addr Instruction" where "Addr" is the value of the PC and "Instruction" is the assembly language instruction pointed to by that PC value. In this mode, all register values except the PC are suppressed.

The optional "SKIP" parameter causes all subroutine calls to be shown as a single step. The subroutine called will be executed but the information shown while stepping will not include the instructions in the subroutine, only the call to it. If the subroutine called, in turn, calls other subroutines, or if the subroutine called contains a large number of instructions, there may be a noticeable wait until the program returns to the main program.

The Until clause is defined as follows:

```
OPERAND1 OPERATION OPERAND2

Where OPERAND1 can be:

Reg <reg> - Register name
Long at <addr> - Long word value at address <addr>
Word at <addr> - Word value at address <addr>
Byte at <addr> - Byte value at address <addr>

OPERAND2 can be:

Reg <reg> - Register name
#<value> - Immediate value (size determined by OPERAND 1)
<addr> - Value at <addr> (size determined by OPERAND 1)

OPERATION can be:

 = - OPERAND1 equal to OPERAND2
== - OPERAND1 equal to OPERAND2
!= - OPERAND1 not equal to OPERAND2
<> - OPERAND1 not equal to OPERAND2
>< - OPERAND1 not equal to OPERAND2
 > - OPERAND1 greater than OPERAND2
 < - OPERAND1 less than OPERAND2
>= - OPERAND1 greater than or equal to OPERAND2
=> - OPERAND1 greater than or equal to OPERAND2
<= - OPERAND1 less than or equal to OPERAND2
=< - OPERAND1 less than or equal to OPERAND2
```

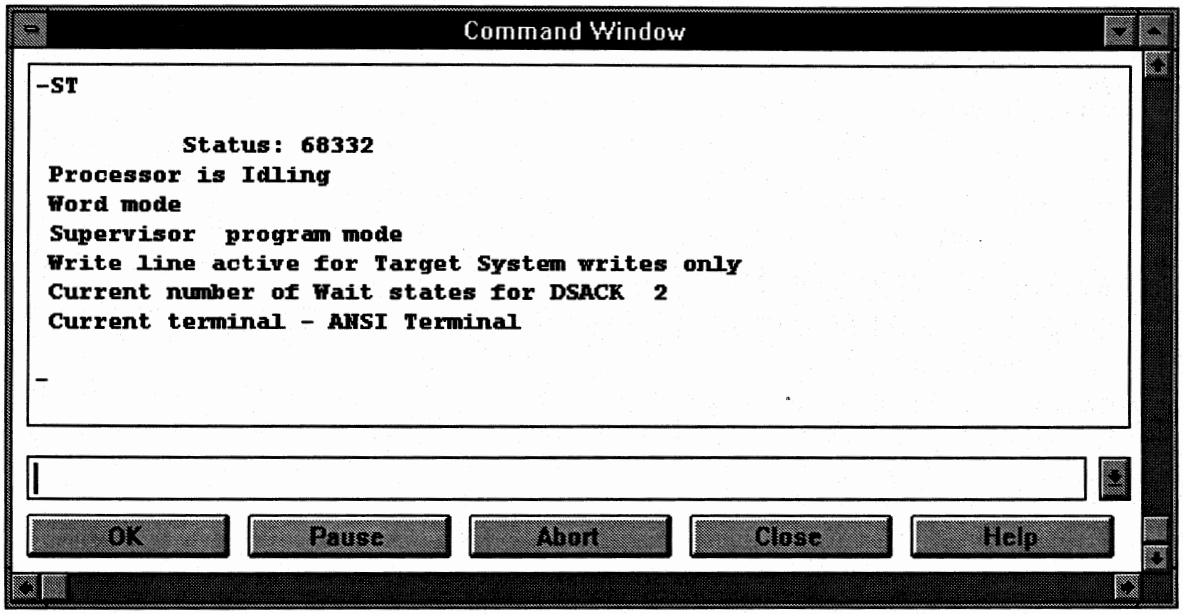
Example:

SS Until word at 5000 = #1234

Would stop stepping when 1234H appears at address 5000H

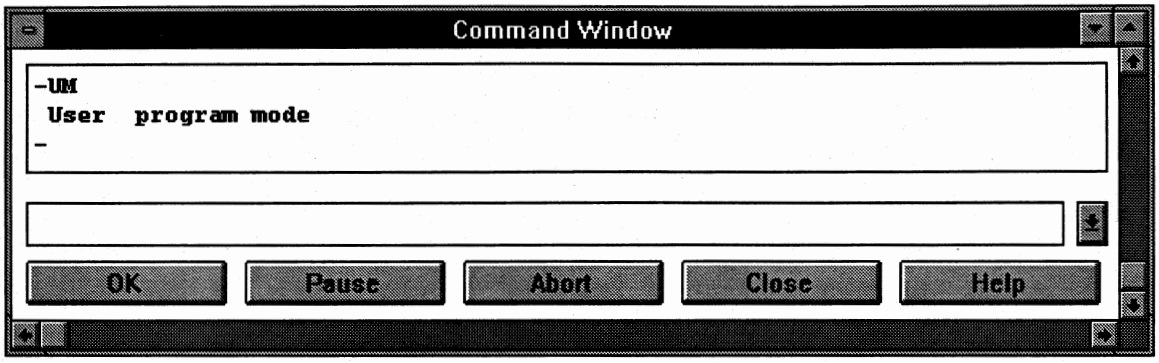
It should be noted that this command takes into account the way data is stored in memory based on the microprocessor manufacturer. If, for example, you wished to stop on a word value of 1234H being in location 5000H, the emulator will look for a 1234H at location 5000H for Motorola processors and 3412H at location 5000H for Intel, Hitachi, and Zilog processors.

ST (Show Emulator Status) Command



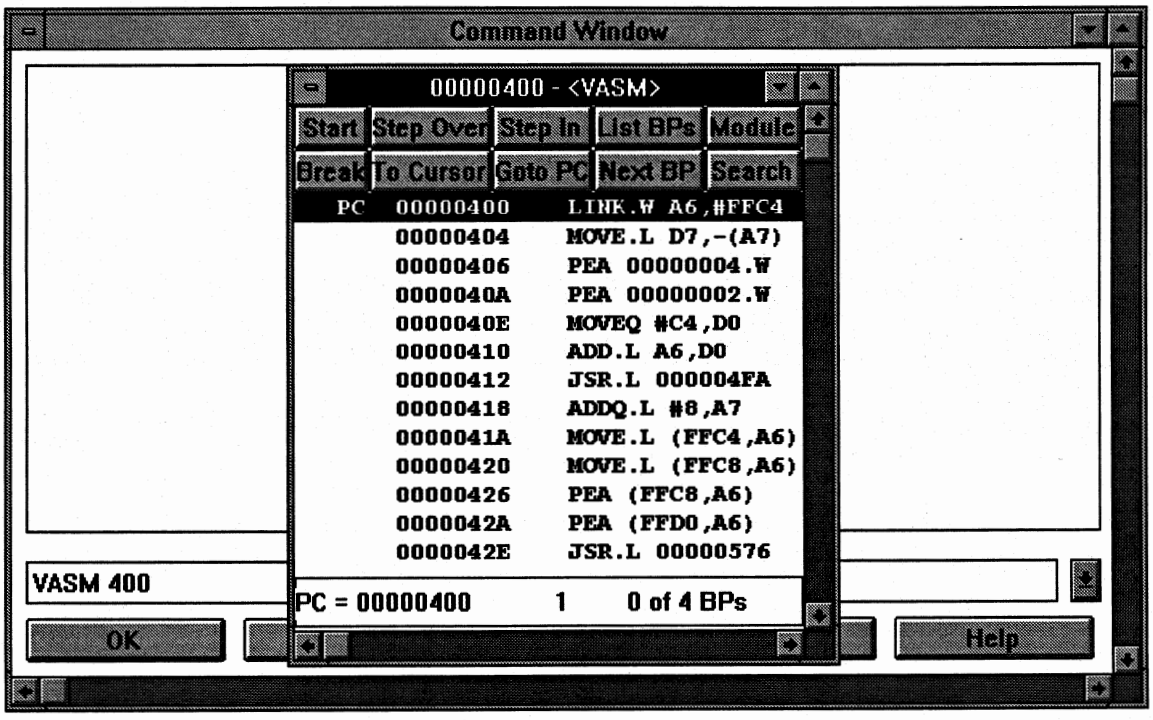
Executing the Status command will display the current status of the emulator.

UM (User Memory Mode) Command



The execution of this command sets the default memory category to be User Memory mode. Using the PM (Program Mode) or DM (Data Mode) commands further define the default memory mode.

VASM (View Assembly) Command



This command will open an assembly Code window beginning at the specified address. **Note that you can also use the 'Source/VASM' menu item to open a VASM window.**

The format of the VASM command is:

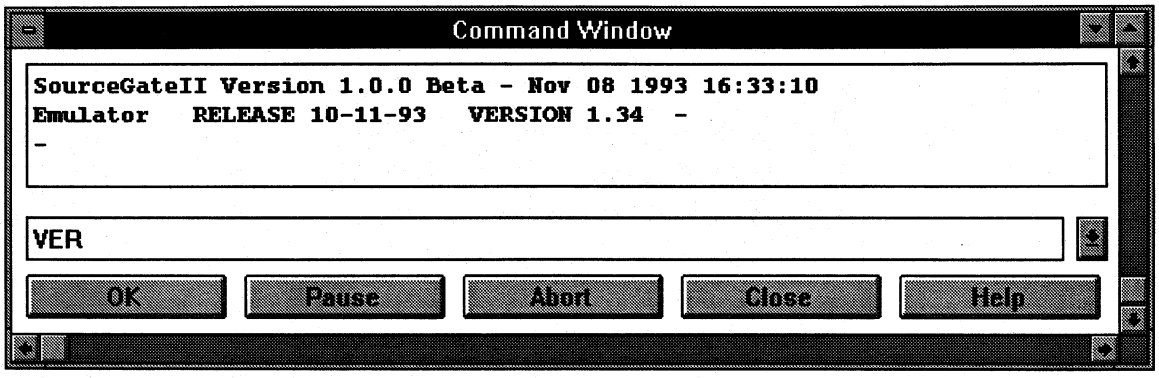
VASM <addr>

Where:

addr: is the beginning address of the VASM window

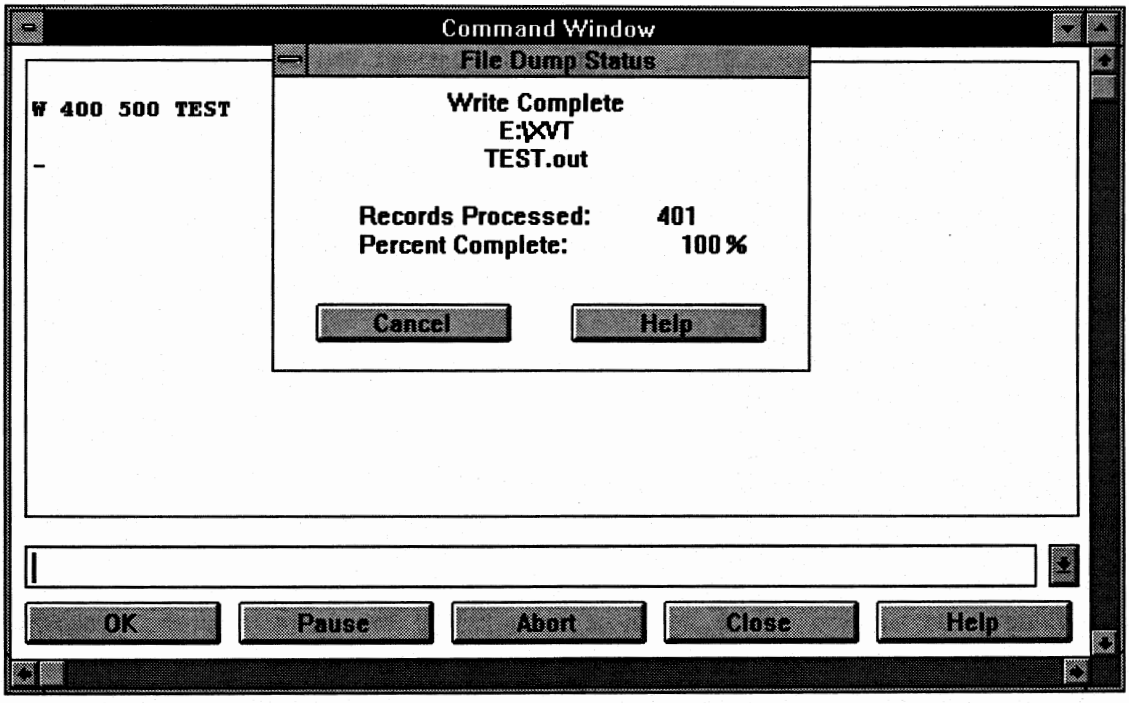
If no address is entered, the VASM window will begin with the address of the current program counter.

VER (Display Firmware Version) Command



This command displays the current version and release date of the emulator firmware.

W/WX (Write Memory Contents To File) Command



This command is used to transfer files from the emulation memory to a host computer. The size of the block being transferred is specified by the two address fields activated by this command. When using the Huntsville Microsystems, Inc. communication software (SourceGate), the block will be written as a Motorola S-Record file with symbol information added to the beginning of the file. This symbol information is in a format that is easily read by HMI software. The extent of this file defaults to .HEX. This symbol/hex file is written to disk on the host computer under the specified filename. **Note that you can also use the 'File/Upload Hex File' menu item to read a file.**

Once the Write Command has been executed, a File Load Status window will appear which will show the status of the write operation. The write operation can be aborted by pressing the **Cancel** button located in the File Load Status window.

The format for writing to the main RS232 port is:

W([SUPDT]) Baddr Eaddr Filename (Flags)

The format for writing to the auxiliary RS232 port is:

WX([SUPDT]) Baddr Eaddr (Flags)

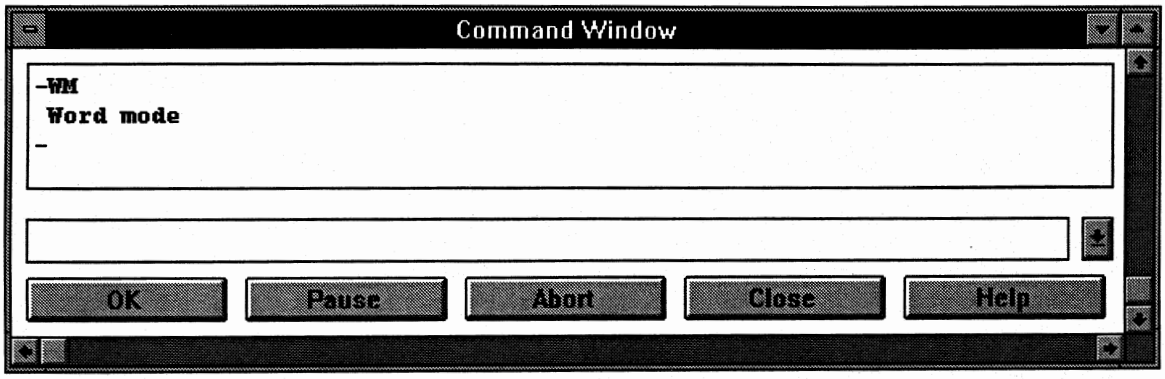
Where:

SUPDT:	are the available overrides
Baddr:	is the hexadecimal beginning address of code to be written
Eaddr:	is the hexadecimal ending address of code to be written
Filename:	is a filename, valid to the host computer, that designates the file to be uploaded (written) to the host computer. If no extension is specified, the default .HEX will be used.
Flags:	are options to be used for each write The options are defined as follows:
H:	Upload/Download only the hex data records present in a hex file. Ignore all symbol records.
S:	Upload/Download only the symbol records of a hex file. Ignore all hex data records.

The 'W' command opens the given file for writing, and writes the contents of the SourceGate internal symbol table into this file, along with the data bytes in memory, from Baddr through Eaddr. 'Baddr' and 'Eaddr' are hexadecimal numbers (or symbols).

Note: The default file extension used on read (load) and write (dump) operations can be changed by modifying the .SG2 file for your emulator. For read operations, the "DefaultLoadExtension" definition can be changed to reflect the desired extension. For write operations, change "DefaultDumpExtension".

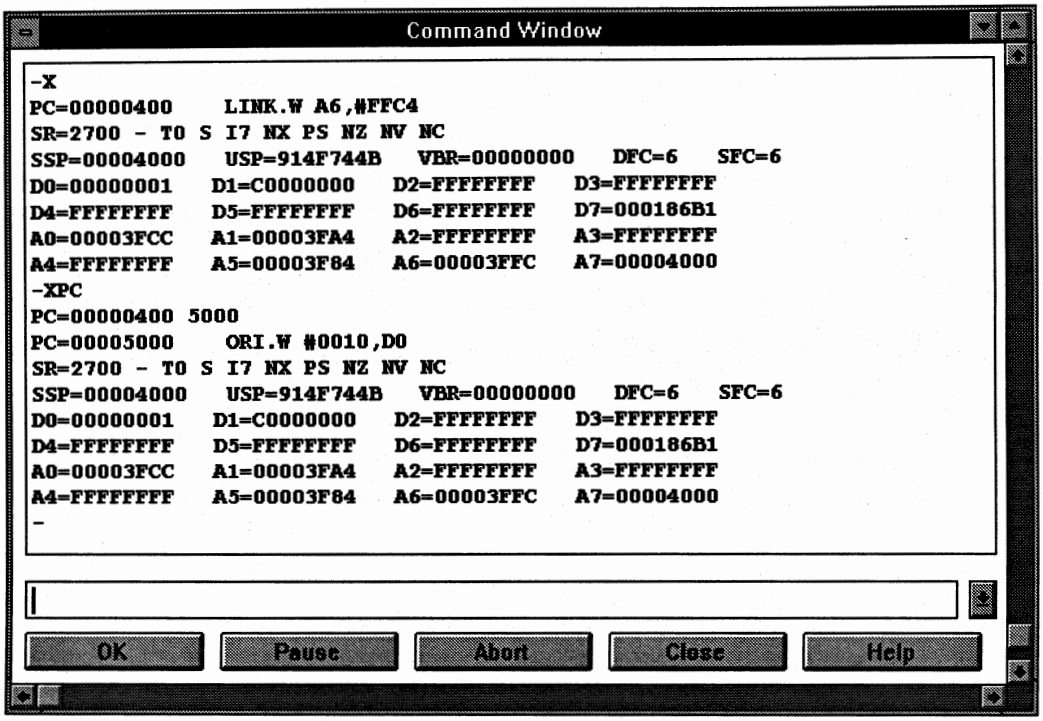
WM (Word Display Mode) Command



This command sets the default for input or output values to "Word Mode". While in Word mode, commands such as D (Dump memory) will display values in Word mode instead of long word mode or byte mode, which are the other choices (see LM and BM).

In addition, commands which require an input value will be affected by this command. For example, if the Fill command, F 8000 9000 33 is executed while in Word mode, each word of memory from 8000H through 9000H will contain 0033H. If this same command was executed with the emulator in long word mode, then 00000033H would be stored in each long word location from 8000H to 9000H. If this same command was executed with the emulator in byte mode, then 33H would be stored in each byte location from 8000H to 9000H.

X (Examine/Change Registers) Command



This command allows you to examine and modify the registers and flags (status register). There are three options to the examine command. The first option is to display all registers and flags. The second option is to display a specific register and allow it to be modified. The last option is to allow the flags to be modified by typing in the condition code status. **Note that you can also use the Register window to display and change the register contents.**

The format for displaying the registers and condition codes is:

X

The format for displaying a specific register and allowing it to be modified is:

XReg

The format for displaying just the condition code values is:

XFL

Where:

Reg: is any processor register such as A1 or SR

When changing a register using the "Xreg" command, you will be shown the current value of the register. To change it, simply type in the new value and then press the "ENTER" key. The register will then contain the new value.

When changing a condition code value using the "XFL" command, you will be shown the current value of the status register and corresponding condition codes. To change condition codes, enter the codes to be changed and then press the "ENTER" key. The register display will then show the modified condition codes.

An explanation of each of the condition codes is listed below:

SR Bit	Flag or Condition Code Name	Not Active Code	Active Code
15	Trace Enable T1	T0 or T1	T2 or T3
14	Trace Enable T0	T0 or T2	T1 or T3
13	Super/User State	U	S
12	Mast/Int State	I	M
11	Not Defined	--	--
10	Int Priority Mask 2		
9	Int Priority Mask 1	I0 through I7	
8	Int Priority Mask 0		
7	Not Defined	--	--
6	Not Defined	--	--
5	Not Defined	--	--
4	Extend	NX	XT
3	Negative	PS	NG
2	Zero	NZ	ZR
1	Overflow	NV	OV
0	Carry	NC	CY

Use these codes, after entering the "XFL" command, to change to a different condition code. For example, if you perform an "XFL" command and are shown the following:

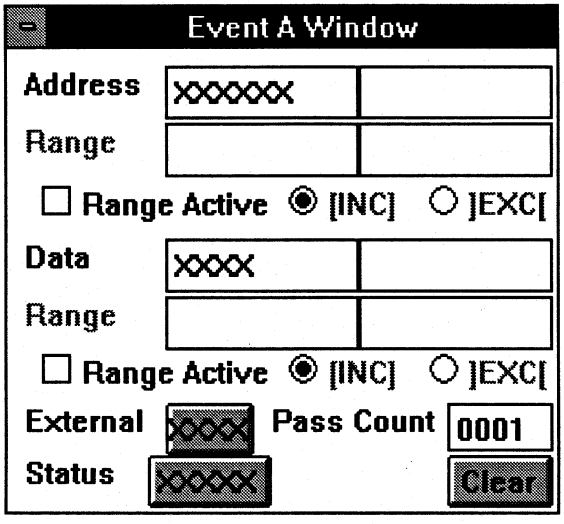
SR = 2709 NT S I7 NX NG NZ NV CY

then you enter "U I6 ZR" and press the "ENTER" key, you will see the following:

SR = 060D NT U I6 NX NG ZR NV CY

The CPU32 Event Windows

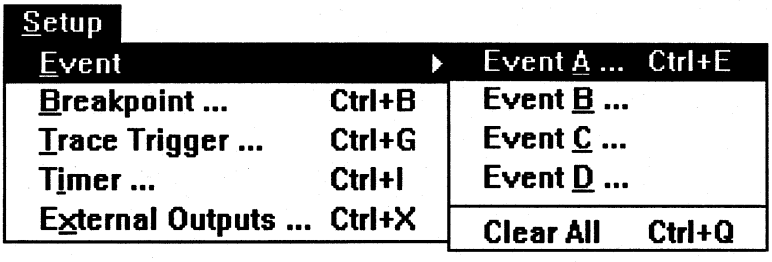
Events are used to define conditions for breakpoints, trigger points, interval timer start and stop conditions, and external output signal conditions. There are four event windows: A, B, C, and D. Each of these events are independent and can be setup for specific conditions. A typical CPU32 Event window has the following format:



Note that 68330/68340 will have a 32-bit address space verses 24-bit for the 68331/68332/68333. An overview of the operation of the Event windows follows:

Opening The Event Windows

To open the Event windows, choose 'Setup/Event'. Click on 'Event' again to see a listing of the four available events:



Click on the desired event in order to open that event window.

Event Window Hexadecimal Fields

The Event windows have several hexadecimal fields that can be defined as part of the event condition. An overview of all possible hexadecimal event fields for the HMI-200-CPU32 emulator is given in the sections that follow. Note that some older versions of this emulator may not support all of these fields. Fields that may not be supported in certain versions of this emulator are so noted in the sections that follow.

Event Address Field
The 'Address' field is used to specify the address to be defined for the event. When this field is defined for an event, the event condition will not become true until the specified address appears on the address bus. Valid values for this field are 0-F and 'X' which is the don't care condition. Binary values can be entered by clicking on the 'Address' field and entering the desired values in the Event Address Binary window. For additional information on entering binary values, see ***Event Window Binary Fields.***

Event Address Range Field
The 'Address Range' field is used to define a range of address as part of the event. For this field to be active, the 'Range Active' box for the 'Address' field must be checked. You should then enter the beginning address of the range in the 'Address' field and the ending address of the range in the 'Address Range' field. When this field is defined for an event, the event condition will not become true until an address that resides within (or outside of) the defined range appears on the address bus. Valid values for this field are 0-F and 'X' which is the don't care condition.

Event Data Field
The 'Data' field is used to specify the data value to be defined for the event. When this field is defined for an event, the event condition will not become true until the specified data value appears on the data bus. Valid values for this field are 0-F and 'X' which is the don't care condition. Binary values can be entered by clicking on the 'Data' field and entering the desired values in the Event Data Binary window. For additional information on entering binary values, see ***Event Window Binary Fields.***

Event Data Range Field *(Not Available On Certain Emulators)*
The 'Data Range' field is used to define a range of data values as part of the event. For this field to be active, the 'Range Active' box for the 'Data' field must be checked. You should then enter the beginning address of the range in the 'Data' field and the ending address of the range in the 'Data Range' field. When this field is defined for an event, the event condition will not become true until a data value that resides within (or outside of) the defined range appears on the data bus. Valid values for this field are 0-F and 'X' which is the don't care condition.

Event Range Active Check Box
The 'Range Active' check box is used to activate address and/or data ranges (if available). For those emulators that support both address and data ranges, there will be a 'Range Active' check box for both. This box must be checked for ranges to be active.

Event Range [INC]/]EXC[Check Boxes
Choosing the [INC] check box will define the range as an inclusive range and the event will become valid when a condition occurs inside of the defined range. Choosing the]EXC[check box will define the range as an exclusive range and the event will become valid when a condition occurs outside of the defined range. For those emulators that support both address and data ranges, there will be a set of [INC]/]EXC[check boxes for both. These fields are only active if the corresponding 'Range Active' box has been checked.

Event Pass Count Field

This field allows you to define the pass count of the event which is the number of times the defined event occurs before it becomes valid. Valid values for this field are 0-F and 'X' which is the don't care condition.

Event Symbol Fields

The fields located to the right of the address and data fields are symbol fields. If there is a symbol name that corresponds with the value entered into the address or data field, it will be displayed here. Also, you can enter a symbol name in these fields by clicking in the field and then pressing the right mouse button. The symbol selection window will then be displayed and symbol names can be selected for the address and data fields. The corresponding hexadecimal value associated with the symbol will be shown in the field to the left of the symbol field. For additional information on using symbols, see ***Working With Symbols*** in Chapter 2.

Event Window Buttons

The following buttons are available in the Event windows.

XXXX (External Bits)

This button will open the Event External Binary window where the external trace bits can be defined as part of an event. For additional information on entering binary values, see ***Event Window Binary Fields.***

XXXX (Status Bits)

This button will open the Event Status Binary window where the status bits can be defined as part of an event. For additional information on entering binary values, see ***Event Window Binary Fields.***

Clear

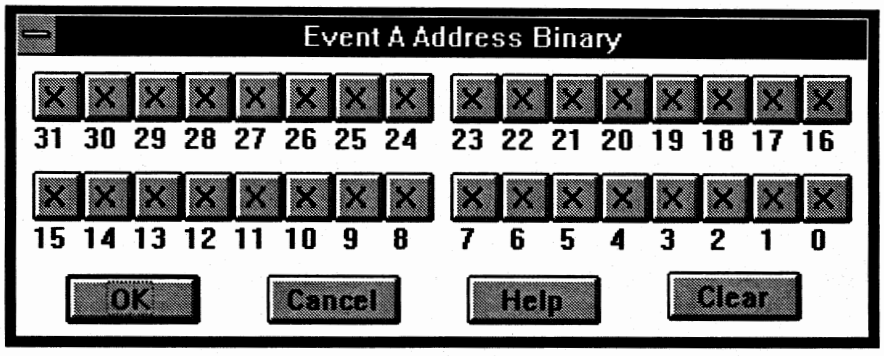
The **Clear** button will set all event fields back to their default values. All hexadecimal fields will contain 'X's (don't cares) and ranges will be disabled.

Event Window Binary Fields

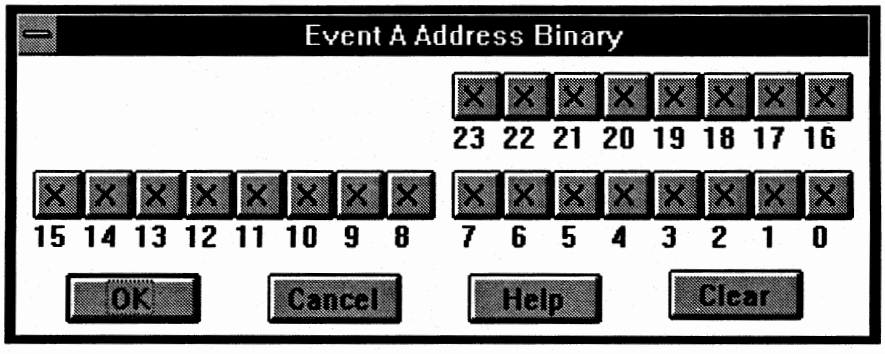
The 'Address', 'Data', 'External', and 'Status' hexadecimal fields have associated binary fields that allow you to define individual bits for these fields. An overview of the various binary event fields is given in the sections that follow.

Event Address Binary Window

If you click on the 'Address' label in the Event window, you will open an 'Event Address Binary' window for that event. This window will look as follows for the 68330/68340:



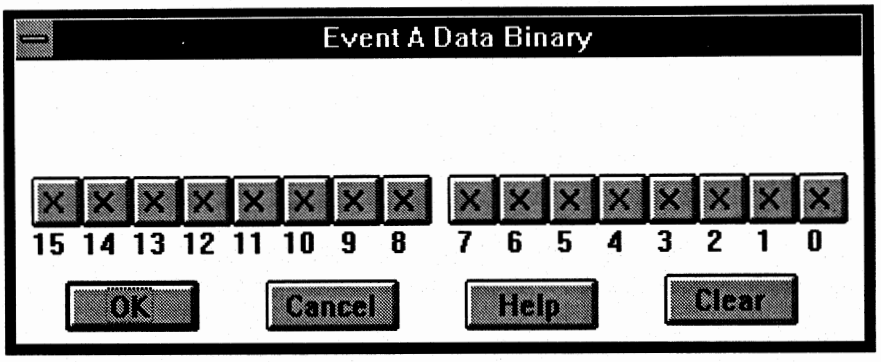
and will look like this for the 68331/68332/68333:



The 'Event Address Binary' window allows you to define an address in binary instead of hexadecimal. To define the event to be valid when an address bit has a particular level on it, place the mouse cursor on the button for the desired address bit and then press the left mouse button. Each press of the left mouse button will toggle through the available choices for the address bit button. You can choose between a "1" for a logic high value, a "0" for a logic low value, and "X" for "Don't Care" which will disable that address bit from influencing the event.

Event Data Binary Window

If you click on the 'Data' label in the Event window, you will open an 'Event Data Binary' window for that event. This window will look as follows:



The 'Event Data Binary' window allows you to define a data value in binary instead of hexadecimal. To define the event to be valid when a data bit has a particular level on it, place the mouse cursor on the button for the desired data bit and then press the left mouse button. Each press of the left mouse button will toggle through the available choices for the data bit button. You can choose between a "1" for a logic high value, a "0" for a logic low value, and "X" for "Don't Care" which will disable that data bit from influencing the event.

Event External Binary Window

If you click on the **External** button in the Event window, you will open an 'Event External' window for that event. This window will look as follows:

The 'Event External' window shows the external trace bits. These bits correspond with the 16 color coded wires that plug into the front of the emulator in the connector labeled "External Trace". These bits can be attached to any TTL level signal and their state can be used to activate an event. They are also shown in the Trace window when one of the State modes is chosen as the display format.

To define the event to be valid when a external trace bit has a particular level on it, place the mouse cursor on the button for the external trace bit and then press the left mouse button. Each press of the left mouse button will toggle through the available choices for the external trace bit button. You can choose between a "1" for a logic high value, a "0" for a logic low value, and "X" for "Don't Care" which will disable the external trace bit from influencing the event.

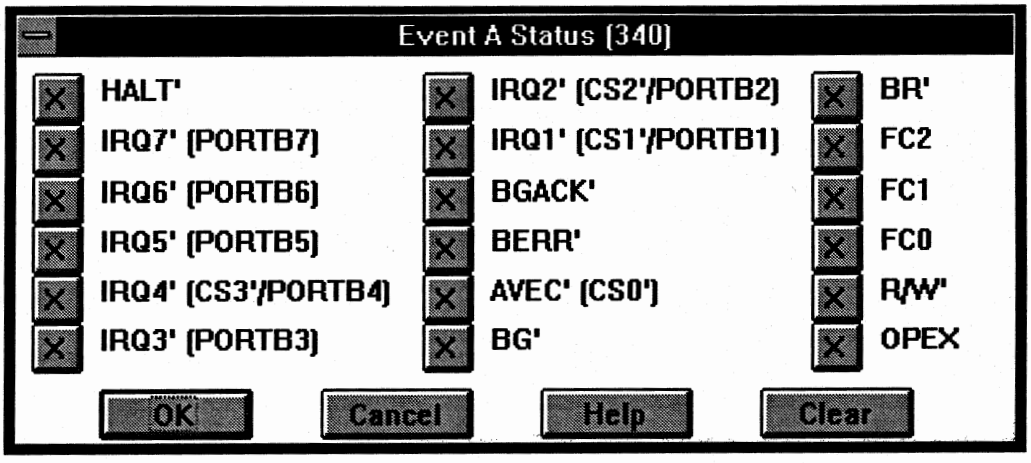
Event Status Binary Window

If you click on the **Status** button in the Event window, you will open an 'Event Status' window for that event. The 'Event Status' window allows you to define a status condition in binary instead of hexadecimal. Status bits are processor specific signals that relate the "status" of the processor. To define the event to be valid when a status bit has a particular level on it, place the mouse cursor on the button for the desired status bit and then press the left mouse button. Each press of the left mouse button will toggle through the available choices for the status bit button. You can choose between a "1" for a logic high value, a "0" for a logic low value, and "X" for "Don't Care" which will disable that status bit from influencing the event.

You may be shown a different status window depending on what emulation pod you are using. Descriptions of all possible configurations are shown in the following sections.

68330/68340:

If you are using one of these processors, you will see the following window:



A description of each of these status bits is given below: Note that any signals that have alternate functions denote the alternate function by enclosing it in brackets.

HALT':

The HALT' status bit allows you to define a condition to be when a HALT condition occurs (HALT'= 0). The inactive state of this signal is (HALT'= 1). An "X" is the don't care condition which means the status bit will be ignored.

IRQ7' thru IRQ1':
The IRQ7'(PORTB7), IRQ6(PORTB6), IRQ5'(PORTB5), IRQ4'(CS3'/PORTB4), IRQ3'(PORTB3), IRQ2'(CS2'/PORTB2), and IRQ1'(CS1'/PORTB1) status bits allow you to define a condition to be when an interrupt is occurring (IRQX'= 0). The inactive state of these signals is (IRQX'= 1). An "X" is the don't care condition which means these status bits will be ignored. Note that () is the alternate function depending on how the pin is programmed.

BGACK':
The BGACK' status bit allows you to define a condition to be when an external device assumes bus mastership (BGACK'=0). The inactive state of this signal is (BGACK'= 1). An "X" is the don't care condition which means the status bit will be ignored.

BERR':
The BERR' status bit allows you to define a condition to be when an invalid bus operation is being attempted, or when used with the HALT signal, that the processor should retry the current cycle (BERR'=0). The inactive state of this signal is (BERR'= 1). An "X" is the don't care condition which means the status bit will be ignored.

AVEC':
The AVEC'(CS0') status bit allows you to define a condition to be when the processor generates an automatic vector during an interrupt acknowledge cycle (AVEC'= 0) or when this signal is not active (AVEC'= 1). An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

BG':
The BG' status bit allows you to define a condition to be when a BG' is asserted in response to a BR'. The inactive state of this signal is (BG'= 1). An "X" is the don't care condition which means the status bit will be ignored.

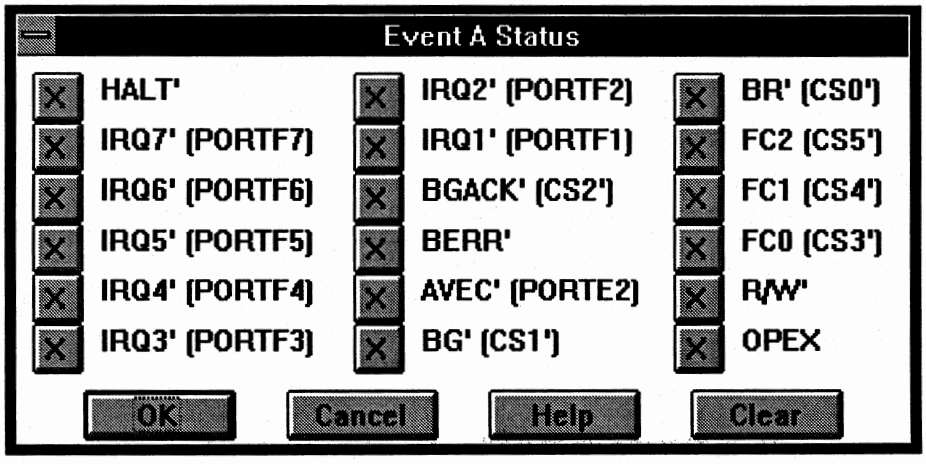
BR':
The BR' status bit allows you to define a condition to be when a BR' is issued. The inactive state of this signal is (BR'= 1). An "X" is the don't care condition which means the status bit will be ignored.

FC0, FC1, and FC2:
The FC0, FC1, and FC2 status bits allow you to define a condition to be when the function codes take on a certain value. An "X" is the don't care condition which means the status bit will be ignored.

R/W':
The R/W' status bit allows you to define a condition to be when a read occurs (R/W'= 1) or when a write occurs (R/W'= 0). An "X" is the don't care condition which means the status bit will be ignored.

OPEX:
The OPEX status bit allows you to define a condition to be when an opcode is executed (OPEX = 1) as opposed to just being fetched and not executed (OPEX = 0). An "X" is the don't care condition which means the status bit will be ignored.

68331/68332/68333:
If you are using one of these processors, you will see the following window:



A description of each of these status bits is given below: Note that any signals that have alternate functions denote the alternate function by enclosing it in brackets.

HALT':
The HALT' status bit allows you to define a condition to be when a HALT condition occurs (HALT'= 0). The inactive state of this signal is (HALT'= 1). An "X" is the don't care condition which means the status bit will be ignored.

IRQ7' thru IRQ1':
The IRQ7'(PORTF7), IRQ6'(PORTF6), IRQ5'(PORTF5), IRQ4'(PORTF4), IRQ3'(PORTF3), IRQ2'(PORTF2), and IRQ1'(PORTF1) status bits allow you to define a condition to be when an interrupt is occurring (IRQX'= 0). The inactive state of these signals is (IRQX'= 1). An "X" is the don't care condition which means these status bits will be ignored. Note that () is the alternate function depending on how the pin is programmed.

BGACK':
The BGACK'(CS2') status bit allows you to define a condition to be when an external device assumes bus mastership (BGACK'=0). The inactive state of this signal is (BGACK'= 1). An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

BERR':
The BERR' status bit allows you to define a condition to be when an invalid bus operation is being attempted, or when used with the HALT signal, that the processor should retry the current cycle (BERR'=0). The inactive state of this signal is (BERR'= 1). An "X" is the don't care condition which means the status bit will be ignored.

AVEC':
The AVEC'(PORTE2) status bit allows you to define a condition to be when the processor generates an automatic vector during an interrupt acknowledge cycle (AVEC'= 0) or when this signal is not active (AVEC'= 1). An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

BG':
The BG'(CS1') status bit allows you to define a condition to be when a BG' is asserted in response to a BR'. The inactive state of this signal is (BG'= 1). An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

BR':
The BR'(CS0') status bit allows you to define a condition to be when a BR' is issued. The inactive state of this signal is (BR'= 1). An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

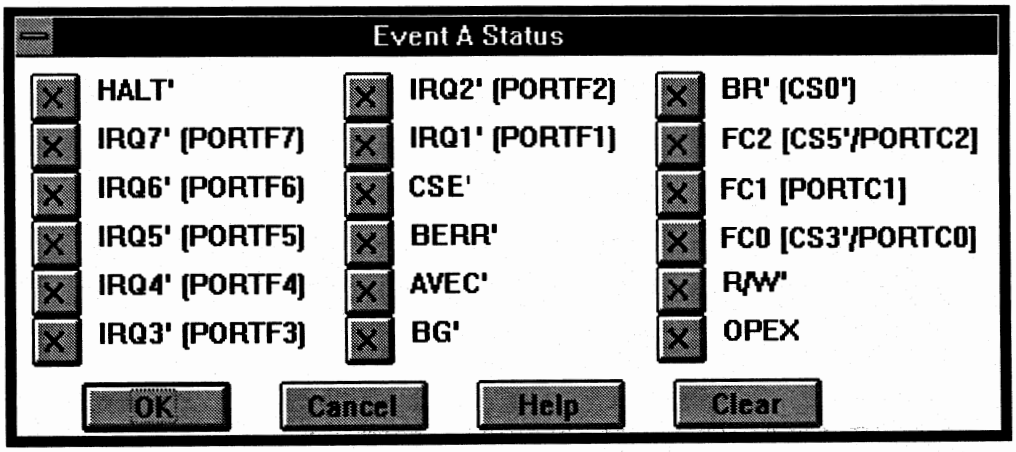
FC0, FC1, and FC2:
The FC0(CS3'),FC1(CS4'), and FC2(CS5') status bits allow you to define a condition to be when the function codes take on a certain value. An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

R/W':
The R/W' status bit allows you to define a condition to be when a read occurs (R/W'= 1) or when a write occurs (R/W'= 0). An "X" is the don't care condition which means the status bit will be ignored.

OPEX:
The OPEX status bit allows you to define a condition to be when an opcode is executed (OPEX = 1) as opposed to just being fetched and not executed (OPEX = 0). An "X" is the don't care condition which means the status bit will be ignored.

Special 68333 Considerations

The status bits shown for the 68333 are the same as are shown for the 68332. However, the 68333 does not support some of the alternate functions of these pins. The following is a listing of the status bits and any available alternate functions, shown enclosed in (), that are supported by the emulator for this processor. These status bits correspond to the positions of the status bits shown in the 68332 status window and should be substituted accordingly:

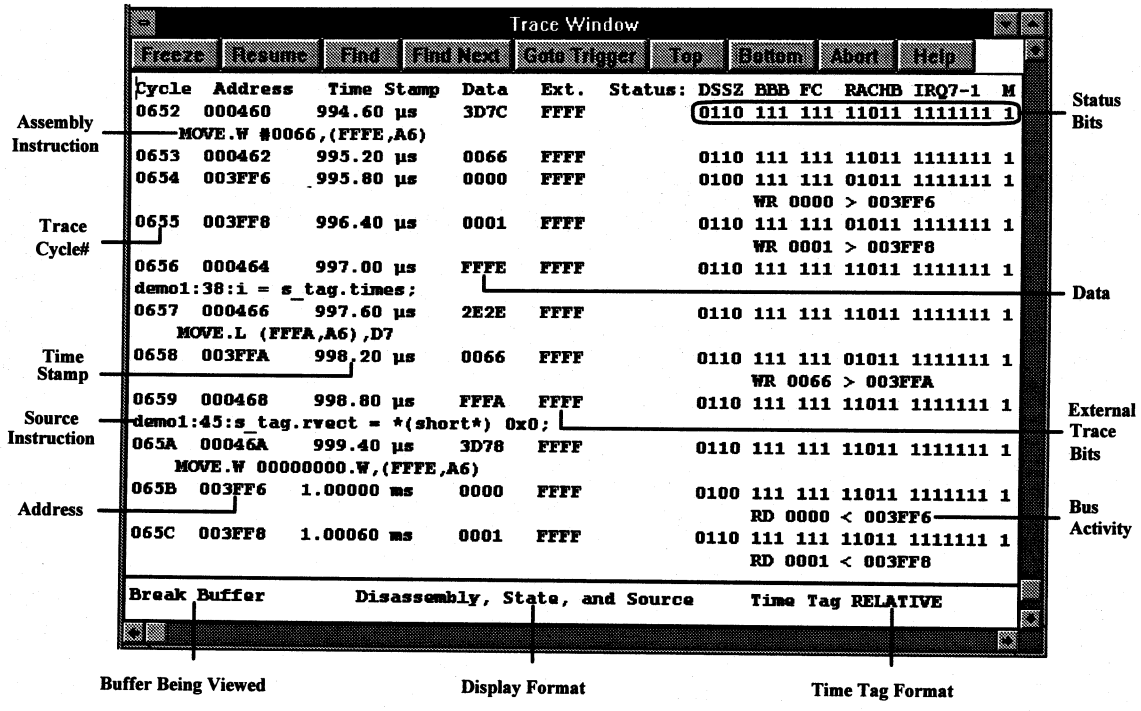


Notice that BGACK' is not available as a status bit since the emulator must use its alternate function, CSE', to talk to the Port Replacement Unit (PRU) located on the pod. The user's code SHOULD NOT program BGACK' to be active as this will cause the processor to lose communication with the PRU.

The CPU32 Trace Window

The Trace window allows you to view and search through the two trace buffers (Trigger and Break) and to initiate and view the Freeze trace. The trace buffers can be displayed in a combination of disassembly, state, source, and time tag information (time tag information requires the Performance Analysis Card (PAC)). Other functions of the Trace window include trace search and trace print.

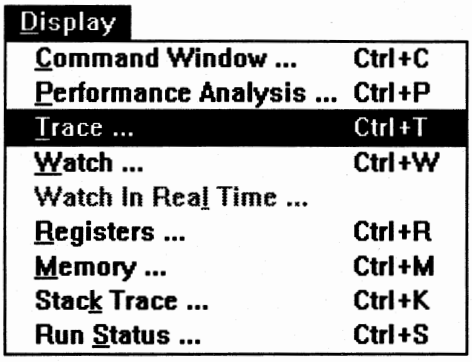
The Trace window is updated each time emulation is terminated through a breakpoint, stop command, reset command, or when a freeze trace command is issued. The information contained in the Trace window is dependent on the emulator being used. An example CPU32 Trace window is shown below.



Up to 16K cycles of trace information can be stored in each buffer in those units that support extended trace. Units not supporting extended trace will store up to 4K cycles in each buffer.

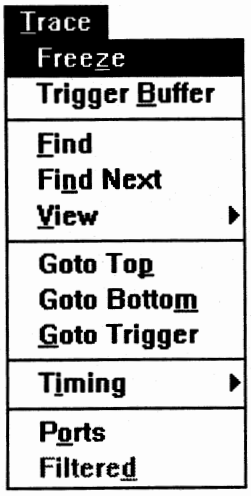
Opening The Trace Window

To open the Trace window, choose 'Display/Trace':



An alternate method of opening the Trace window is to use the accelerator key combination Ctrl+T.

When the trace window is open, an additional menu item labeled "Trace" is shown in the SourceGate II menu bar. Selecting this menu item will show you an additional set of selections which allow you to define what information is shown in the trace buffers:



An overview of each of these menu items is given in the sections that follow.

Trace Window Buttons

A description of the functions of the buttons available in this window are as follows:

Freeze

The **Freeze** button allows you to view the trace and to modify events and sequences while the emulator continues to run.

Resume

The **Resume** button should be pressed when the trace is in freeze mode in order to return to the normal mode of emulation.

Find

The **Find** button should be pressed when you wish to search for a particular pattern in the trace buffer.

Find Next

The **Find Next** button will find the next occurrence of the defined search pattern.

Goto Trigger

The **Goto Trigger** button will find the trace trigger if it exists in the trace buffer.

Top

The **Top** button will take the display to the top of the trace buffer.

Bottom

The **Bottom** button will take the display to the bottom of the trace buffer.

Abort

The **Abort** button will stop the trace display from scrolling after the current scroll operation is completed. This button is handy if you have entered multiple page up or page down operations and then decide to cancel the scrolling operation.

Help

The **Help** button will invoke help for this window.

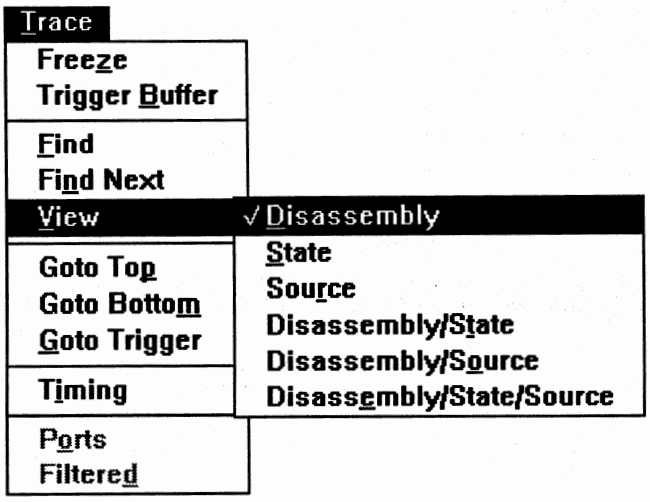
Choosing The Trace Buffer To View

Since the emulator has both a 'Break' and a 'Trigger' buffer, you can specify which of these buffers to view in the Trace window. This is done by choosing 'View/Break' to view the Break buffer or 'View/Trigger' to view the Trigger buffer. Note that the Break buffer will always be available but, for the Trigger buffer to exist, a valid trace trigger condition along with the specified trace delay as defined in the 'Trace Trigger' window must have occurred. If these conditions have not occurred, then choosing the Trigger buffer will produced a message that states "No Data Recorded".

Also note that the buffer selection shown in the Trace menu depends upon the current buffer being viewed. If the Break Buffer is being viewed, the menu will show "Trigger Buffer" as a command. If the Trigger Buffer is being viewed, the menu will show "Break Buffer" as a command.

Trace Display Modes

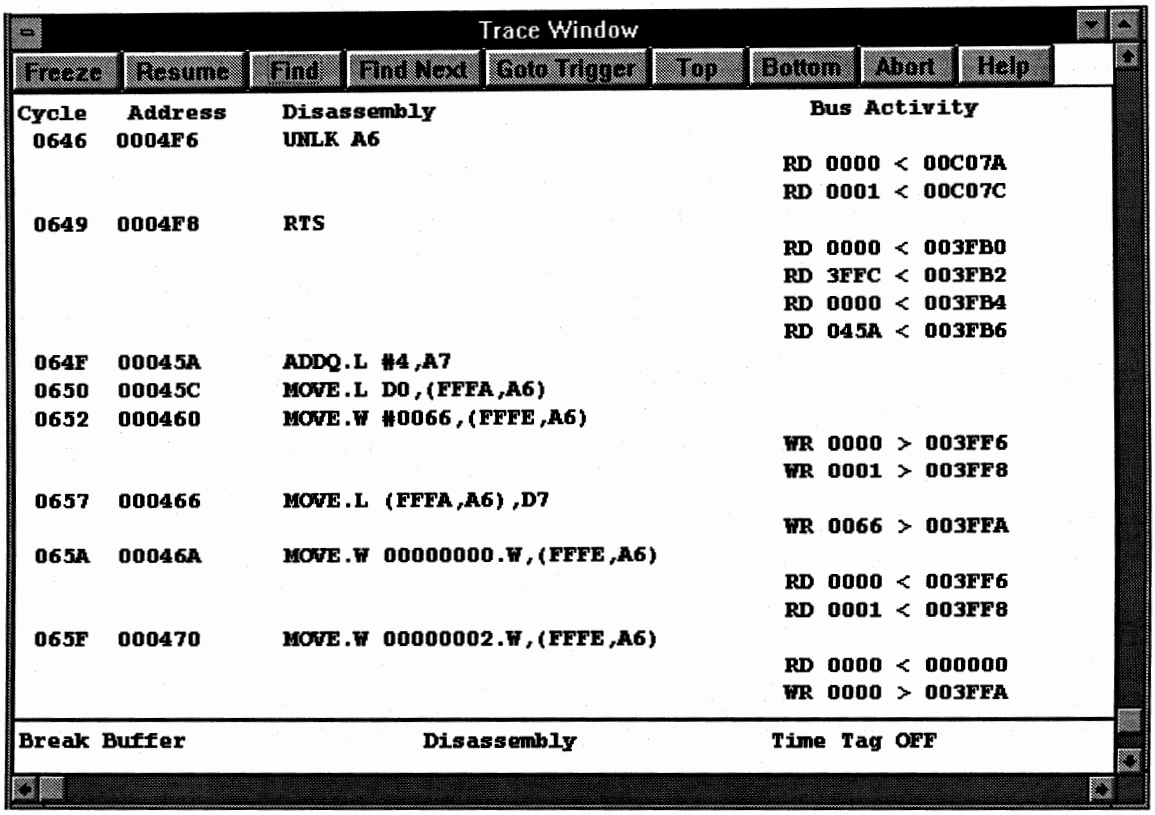
The display format of the trace display can be chosen by using one or a combination of the "View" options. The available options will be shown in a submenu with the following format:



An explanation of each of these options follows:

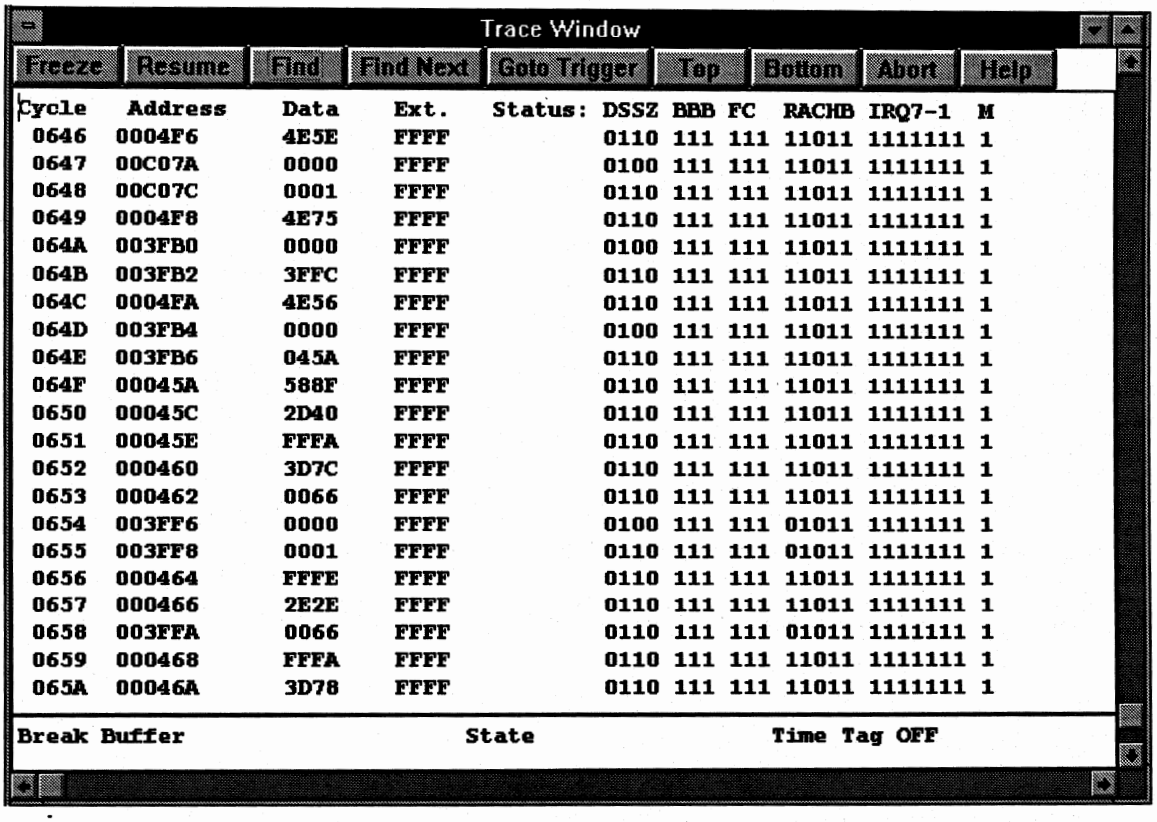
Disassembly

The trace buffers will show only disassembly information which includes cycle, address, disassembled instructions, bus activity, and time stamp (if applicable). All state and source code information will be suppressed. To select this mode, choose 'Trace/View/Disassembly'. An example CPU32 Trace window is shown below in 'Disassembly' format:



State

The trace buffers will show only state information which includes cycle, address, data, external trace bits, status bits, bus activity, and time stamp (if applicable). All disassembly and source code information will be suppressed. To select this mode, choose 'Trace/View/State'. An example CPU32 Trace window is shown below in 'State' format:



Cycle	Address	Data	Ext.	Status:	DSSZ	BBB	FC	RACHB	IRQ7-1	M
0646	0004F6	4E5E	FFFF		0110	111	111	11011	1111111	1
0647	00C07A	0000	FFFF		0100	111	111	11011	1111111	1
0648	00C07C	0001	FFFF		0110	111	111	11011	1111111	1
0649	0004F8	4E75	FFFF		0110	111	111	11011	1111111	1
064A	003FB0	0000	FFFF		0100	111	111	11011	1111111	1
064B	003FB2	3FFC	FFFF		0110	111	111	11011	1111111	1
064C	0004FA	4E56	FFFF		0110	111	111	11011	1111111	1
064D	003FB4	0000	FFFF		0100	111	111	11011	1111111	1
064E	003FB6	045A	FFFF		0110	111	111	11011	1111111	1
064F	00045A	588F	FFFF		0110	111	111	11011	1111111	1
0650	00045C	2D40	FFFF		0110	111	111	11011	1111111	1
0651	00045E	FFFA	FFFF		0110	111	111	11011	1111111	1
0652	000460	3D7C	FFFF		0110	111	111	11011	1111111	1
0653	000462	0066	FFFF		0110	111	111	11011	1111111	1
0654	003FF6	0000	FFFF		0100	111	111	01011	1111111	1
0655	003FF8	0001	FFFF		0110	111	111	01011	1111111	1
0656	000464	FFFE	FFFF		0110	111	111	11011	1111111	1
0657	000466	2E2E	FFFF		0110	111	111	11011	1111111	1
0658	003FFA	0066	FFFF		0110	111	111	01011	1111111	1
0659	000468	FFFA	FFFF		0110	111	111	11011	1111111	1
065A	00046A	3D78	FFFF		0110	111	111	11011	1111111	1

The status bits shown in the trace buffer are defined as follows when using a 68330 or 68340 processor:

DSSZ	DSACK1', DSACK0', SIZ1, SIZ0
BBB	BR', BG', BGACK'
FC	FC2, FC1, FC0
RACHB	R/W', AVEC'(CS0'), CS3'(IRQ4'/PORTB4), HALT', BERR'
IRQ7-1	IRQ7'(PORTB7), IRQ6'(PORTB6), IRQ5'(PORTB5), IRQ4'(CS3'/PORTB4), IRQ3'(PORTB3), IRQ2'(CS2'/PORTB2), IRQ1'(CS1'/PORTB1)
M	MODCK(PORTB0)

Note: () is alternate function depending on how pin is programmed

The status bits shown in the trace buffer are defined as follows when using a 68331 or 68332 processor:

DSSZ	DSACK1'(PORTE1), DSACK0'(PORTE0), SIZ1(PORTE7), SIZ0(PORTE6)
BBB	BR'(CS0'), BG'(CS1'), BGACK'(CS2')
FC	FC2(CS5'), FC1(CS4'), FC0(CS3')
RACHB	R/W', AVEC'(PORTE2), CSBOOT', HALT', BERR'
IRQ7-1	IRQ7'(PORTF7), IRQ6'(PORTF6), IRQ5'(PORTF5), IRQ4'(PORTF4), IRQ3'(PORTF3), IRQ2'(PORTF2), IRQ1'(PORTF1)
M	MODCK(PORTF0)

Note: () is alternate function depending on how pin is programmed

The status bits shown in the trace buffer are defined as follows when using a 68333 processor:

DSSZ	DSACK1'. DSACK0', SIZ1, SIZ0
BBB	BR'(CS0'), BG', CSE'
FC	FC2(CS5'/PORTC2), FC1(PORTC1), FC0(CS3'/PORTC0)
RACHB	R/W', AVEC', CSBOOT', HALT', BERR'
IRQ7-1	IRQ7'(PORTF7), IRQ6'(PORTF6), IRQ5'(PORTF5), IRQ4'(PORTF4), IRQ3'(PORTF3), IRQ2'(PORTF2), IRQ1'(PORTF1)
M	MODCK(PORTF0)

Note: () is alternate function depending on how pin is programmed

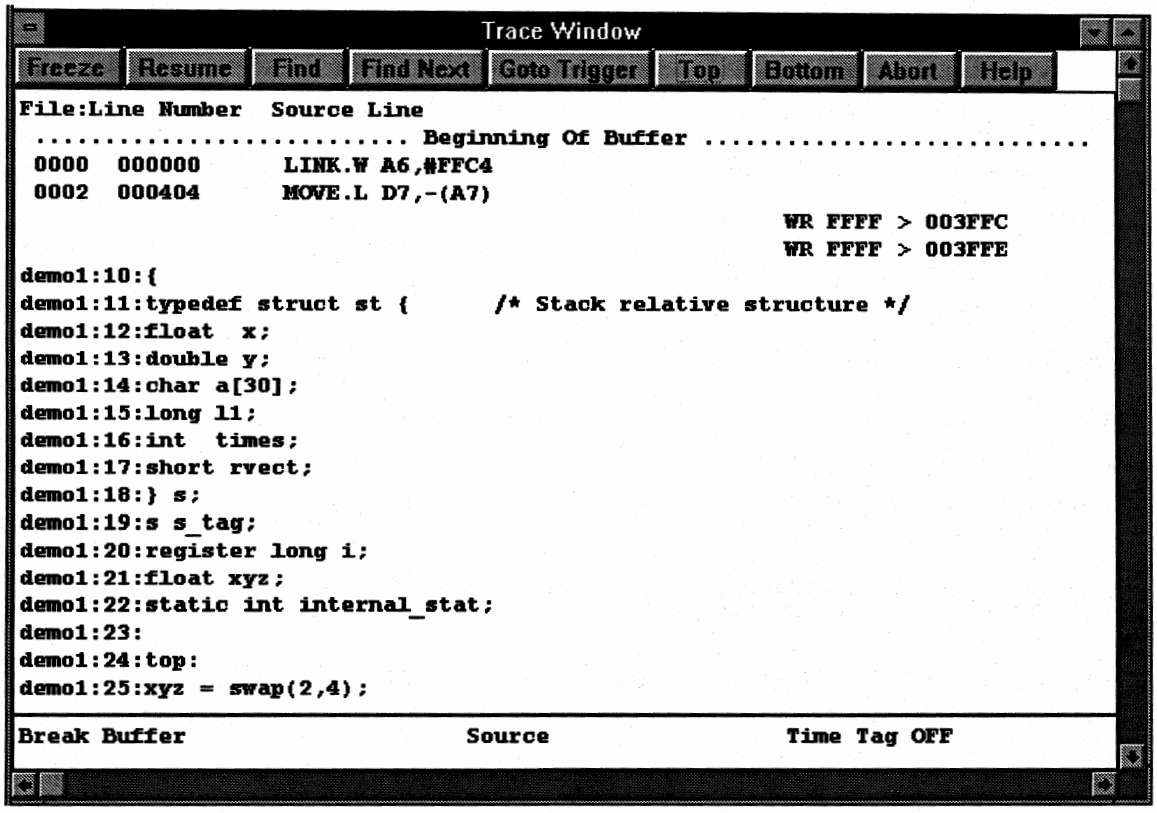
See ***Trace Search Status Bits*** for more detailed information on these status bits.

Source

The trace buffers will show only high-level source information which includes the module name, line number of the module, the actual high-level line of code, and time stamp (if applicable). To select this mode, choose 'Trace/View/Source'.

It should be noted that some disassembled instructions and bus activity may also be shown while in this mode. This information usually appears when compiler routines are traced that were written in assembly language and, thus, have no high- level source code available for them. SourceGate has no choice but to display this information as the "source" associated with the displayed cycles.

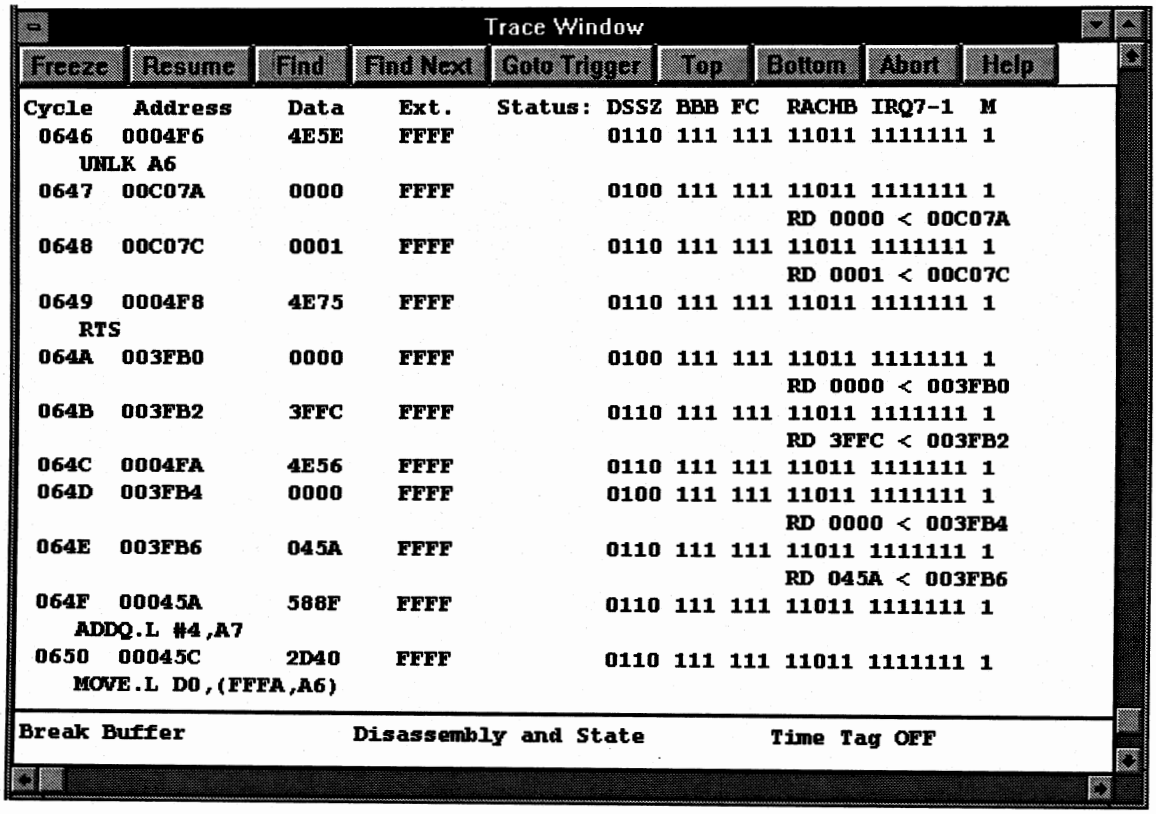
An example CPU32 Trace window is shown below in 'Source' format:



Disassembly/State

The trace buffers will show both disassembly and state information which includes cycle, address, data, external trace bits, status bits, bus activity, and time stamp (if applicable). All high-level source information will be suppressed. To select this mode, choose 'Trace/View/Disassembly/State'.

An example CPU32 Trace window is shown below in 'Disassembly/State' format:



The status bits shown in the trace buffer are defined as follows when using a 68330 or 68340 processor:

DSSZ	DSACK1', DSACK0', SIZ1, SIZ0
BBB	BR', BG', BGACK'
FC	FC2, FC1, FC0
RACHB	R/W', AVEC'(CS0'), CS3'(IRQ4'/PORTB4), HALT', BERR'
IRQ7-1	IRQ7'(PORTB7), IRQ6'(PORTB6), IRQ5'(PORTB5), IRQ4'(CS3'/PORTB4), IRQ3'(PORTB3), IRQ2'(CS2'/PORTB2), IRQ1'(CS1'/PORTB1)
M	MODCK(PORTB0)

Note: () is alternate function depending on how pin is programmed

The status bits shown in the trace buffer are defined as follows when using a 68331 or 68332 processor:

DSSZ	DSACK1'(PORTE1), DSACK0'(PORTE0), SIZ1(PORTE7), SIZ0(PORTE6)
BBB	BR'(CS0'), BG'(CS1'), BGACK'(CS2')
FC	FC2(CS5'), FC1(CS4'), FC0(CS3')
RACHB	R/W', AVEC'(PORTE2), CSBOOT', HALT', BERR'
IRQ7-1	IRQ7'(PORTF7), IRQ6'(PORTF6), IRQ5'(PORTF5), IRQ4'(PORTF4), IRQ3'(PORTF3), IRQ2'(PORTF2), IRQ1'(PORTF1)
M	MODCK(PORTF0)

Note: () is alternate function depending on how pin is programmed

The status bits shown in the trace buffer are defined as follows when using a 68333 processor:

DSSZ	DSACK1'. DSACK0', SIZ1, SIZ0
BBB	BR'(CS0'), BG', CSE'
FC	FC2(CS5'/PORTC2), FC1(PORTC1), FC0(CS3'/PORTC0)
RACHB	R/W', AVEC', CSBOOT', HALT', BERR'
IRQ7-1	IRQ7'(PORTF7), IRQ6'(PORTF6), IRQ5'(PORTF5), IRQ4'(PORTF4), IRQ3'(PORTF3), IRQ2'(PORTF2), IRQ1'(PORTF1)
M	MODCK(PORTF0)

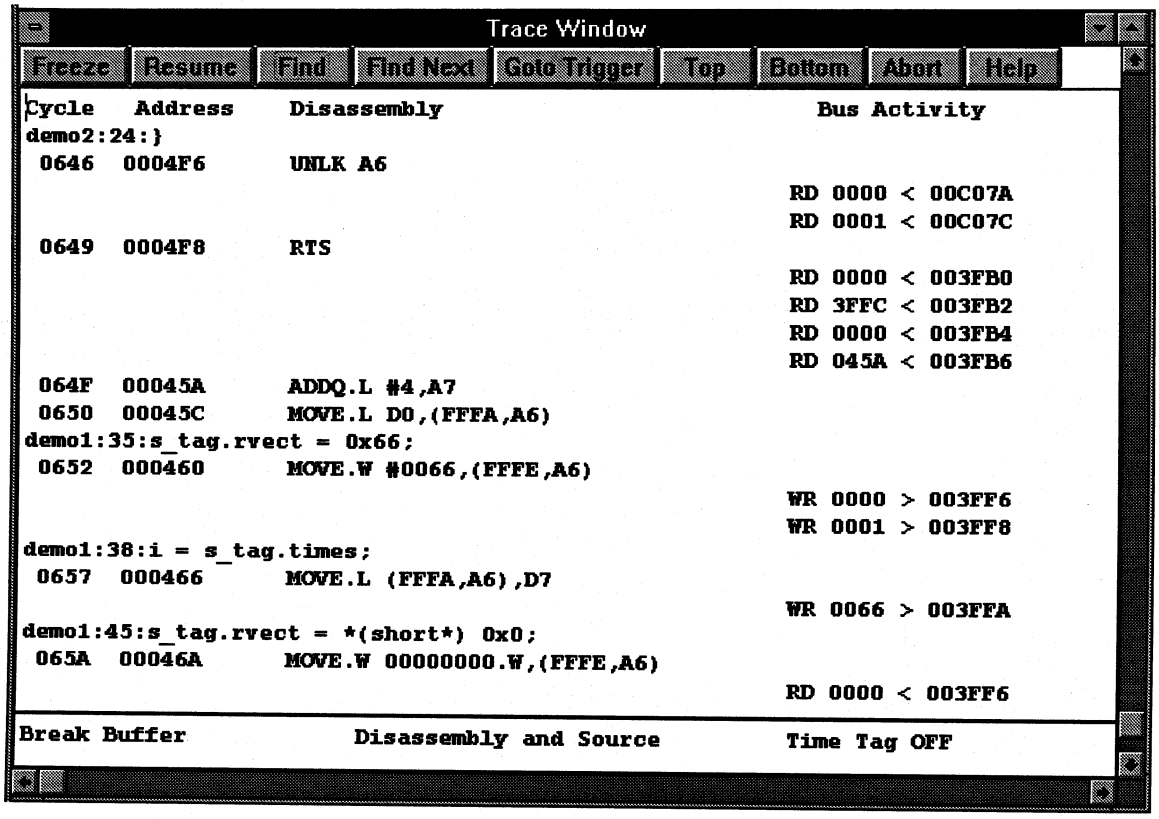
Note: () is alternate function depending on how pin is programmed

See ***Trace Search Status Bits*** for more detailed information on these status bits.

Disassembly/Source

The trace buffers will show both disassembly and source information which includes cycle, address, module name, line number of the module, the actual high-level line of code, bus activity, and time stamp (if applicable). All state specific information will be suppressed. To select this mode, choose 'Trace/View/Disassembly/Source'.

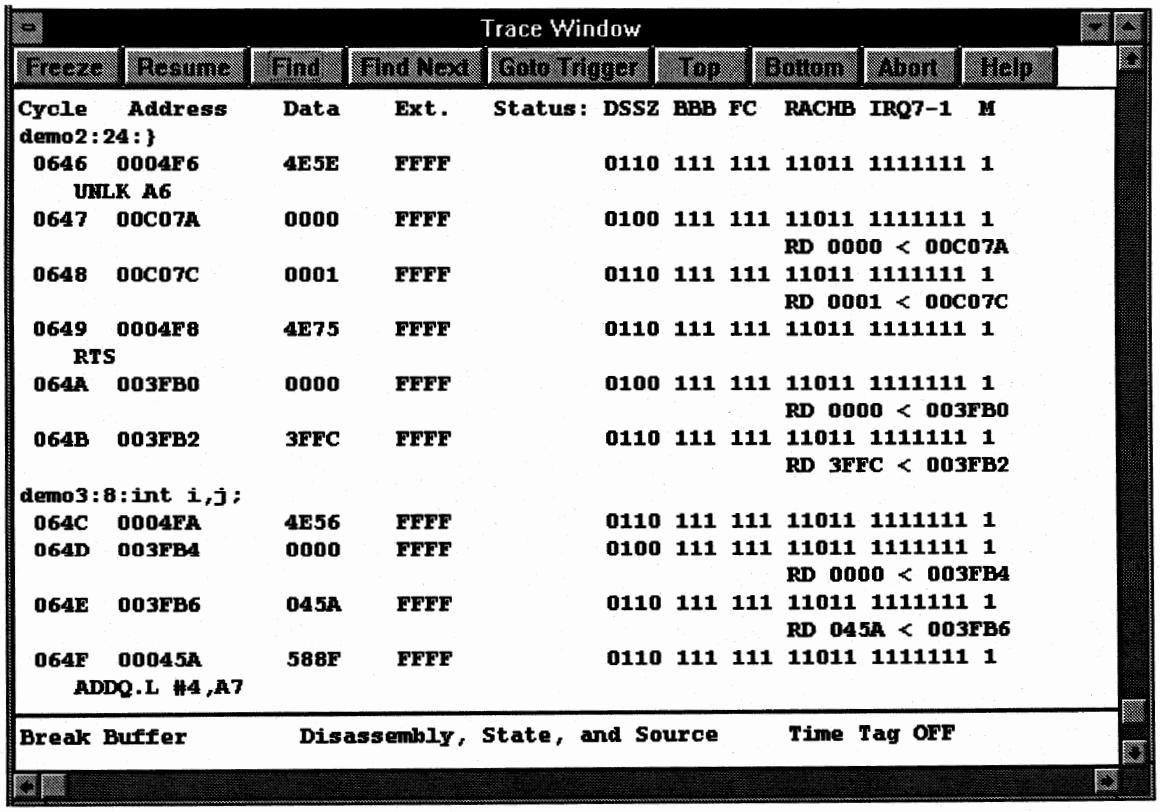
An example CPU32 Trace window is shown below in 'Disassembly/Source' format:



Disassembly/State/Source

The trace buffers will show all available information while in this mode which includes cycle, address, data, external trace bits, status bits, bus activity, module name, line number of the module, the actual high-level line of code, and time stamp (if applicable). No information is suppressed in this mode. To select this mode, choose 'Trace/View/Disassembly/State/Source'.

An example CPU32 Trace window is shown below in 'Disassembly/State/Source' format:



The status bits shown in the trace buffer are defined as follows when using a 68330 or 68340 processor:

DSSZ	DSACK1', DSACK0', SIZ1, SIZ0
BBB	BR', BG', BGACK'
FC	FC2, FC1, FC0
RACHB	R/W', AVEC'(CS0'), CS3'(IRQ4'/PORTB4), HALT', BERR'
IRQ7-1	IRQ7'(PORTB7), IRQ6'(PORTB6), IRQ5'(PORTB5), IRQ4'(CS3'/PORTB4), IRQ3'(PORTB3), IRQ2'(CS2'/PORTB2), IRQ1'(CS1'/PORTB1)
M	MODCK(PORTB0)

Note: () is alternate function depending on how pin is programmed

The status bits shown in the trace buffer are defined as follows when using a 68331 or 68332 processor:

DSSZ	DSACK1'(PORTE1), DSACK0'(PORTE0), SIZ1(PORTE7), SIZ0(PORTE6)
BBB	BR'(CS0'), BG'(CS1'), BGACK'(CS2')
FC	FC2(CS5'), FC1(CS4'), FC0(CS3')
RACHB	R/W', AVEC'(PORTE2), CSBOOT', HALT', BERR'
IRQ7-1	IRQ7'(PORTF7), IRQ6'(PORTF6), IRQ5'(PORTF5), IRQ4'(PORTF4), IRQ3'(PORTF3), IRQ2'(PORTF2), IRQ1'(PORTF1)
M	MODCK(PORTF0)

Note: () is alternate function depending on how pin is programmed

The status bits shown in the trace buffer are defined as follows when using a 68333 processor:

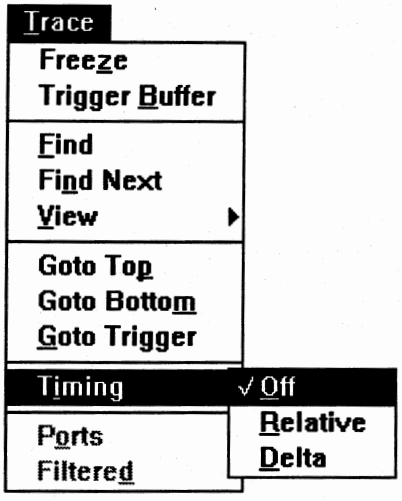
DSSZ	DSACK1'. DSACK0', SIZ1, SIZ0
BBB	BR'(CS0'), BG', CSE'
FC	FC2(CS5'/PORTC2), FC1(PORTC1), FC0(CS3'/PORTC0)
RACHB	R/W', AVEC', CSBOOT', HALT', BERR'
IRQ7-1	IRQ7'(PORTF7), IRQ6'(PORTF6), IRQ5'(PORTF5), IRQ4'(PORTF4), IRQ3'(PORTF3), IRQ2'(PORTF2), IRQ1'(PORTF1)
M	MODCK(PORTF0)

Note: () is alternate function depending on how pin is programmed

See ***Trace Search Status Bits*** for more detailed information on these status bits.

Using The Time Stamp Feature

If you have the optional Performance Analysis Card (PAC) installed in your emulator, a time stamp can be included for each cycle in the trace display. The format of this time stamp can be selected by using one or a combination of the "Timing" options. The available options will be shown in a submenu with the following format:



An explanation of each of these timing formats is given in the sections that follow. **You must have the PAC option installed in your system or the "Relative" and "Delta" options will be greyed out. No timing values will be shown in the Trace window unless the PAC option is installed.**

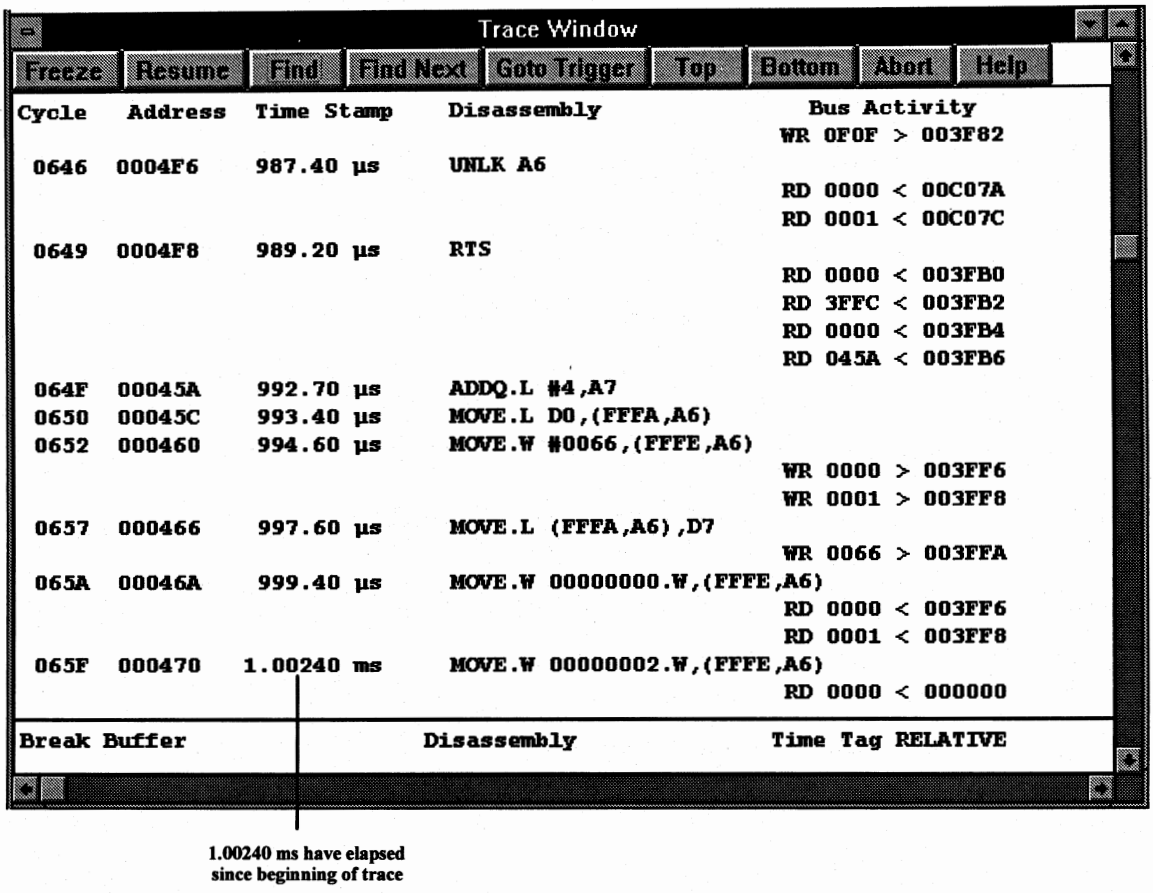
Timing Off

If the optional Performance Analysis Card (PAC) is installed in the emulator, a time stamp can be included for each cycle in the trace display. This option turns the time stamp off.

Timing Relative

If the optional Performance Analysis Card (PAC) is installed in the emulator, a time stamp can be included for each cycle in the trace display. To select this mode, choose 'Trace/Timing/Relative'. This option makes the time stamp to be relative to the beginning of the trace buffer.

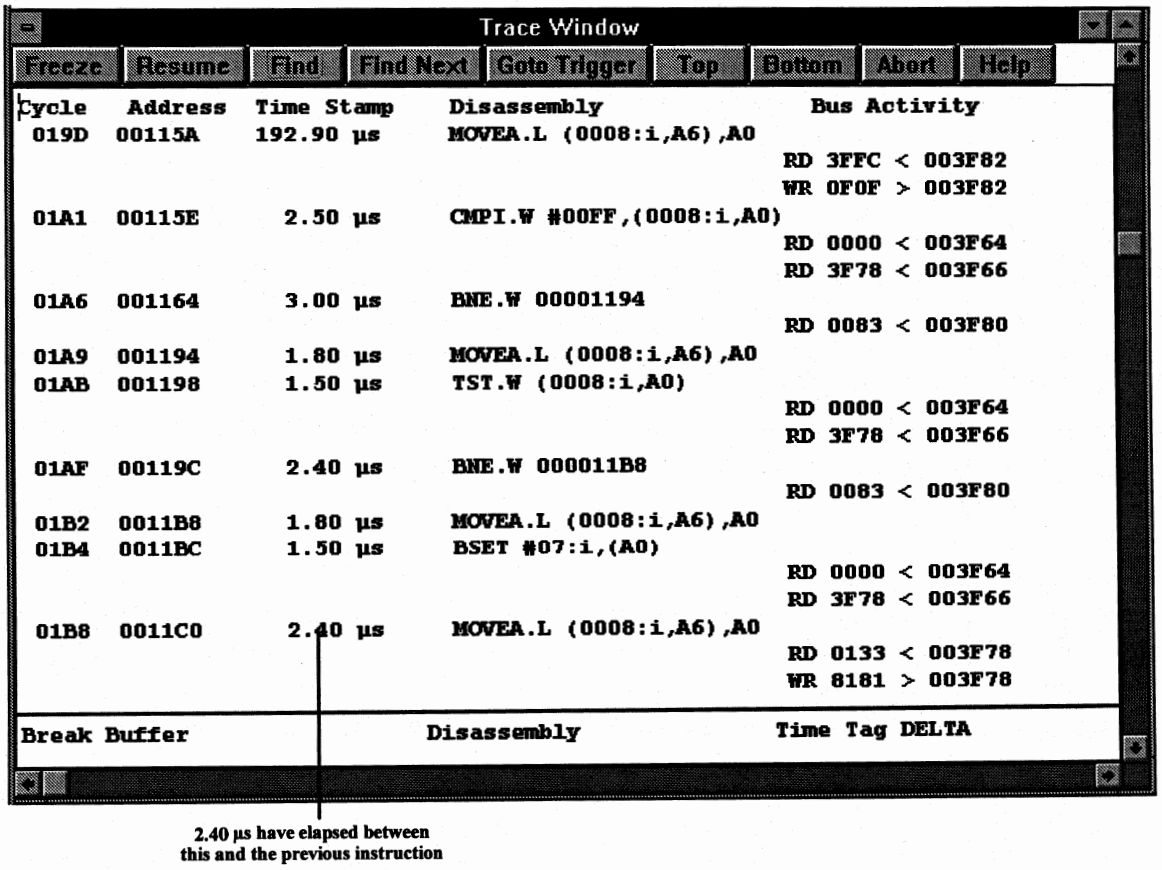
An example CPU32 Trace window is shown below with the time stamp set to be 'Relative':



Timing Delta

If the optional Performance Analysis Card (PAC) is installed in the emulator, a time stamp can be included for each cycle in the trace display. To select this mode, choose 'Trace/Timing/Delta'. This option makes the time stamp to be a delta value between each cycle.

An example CPU32 Trace window is shown below with the time stamp set to be 'Delta':



Freeze Trace Operation

The 'Freeze' feature allows you to view the trace buffers without stopping emulation. Also, while you are in this 'Freeze Trace' mode of operation, the events, breakpoint conditions, etc. can be changed. When the 'Resume' command is issued, the emulator will be reprogrammed with the changed parameters. This allows you to view the trace information and, if you don't like what you see or if your defined breakpoint condition, trigger condition, etc. is not occurring correctly, you can change the conditions so that you get the result that you want, all of this without having to disturb the real-time emulation of the processor. This feature can prove to be invaluable, in the embedded world, to those designers that have complex target systems that require elaborate startup procedures. The following sections give an overview of how to use the Freeze/Resume feature.

Freezing The Trace

The 'Freeze' command can be issued in one of three ways. From the Trace window, you can either choose 'Trace/Freeze' or press the 'Freeze' button. The 'Freeze' command can also be issued from the 'Run Status' window. After the 'Freeze' command is issued, you can then view the trace buffers and change the event, breakpoint, trace trigger, interval timer, and external output conditions. If the trace trigger and associated delay condition have occurred, the freeze trace will be the same information as contained in the trigger trace. If these conditions have not occurred, the freeze trace will show a 'snapshot' of the last 4K cycles of information executed before the 'Freeze' command was given.

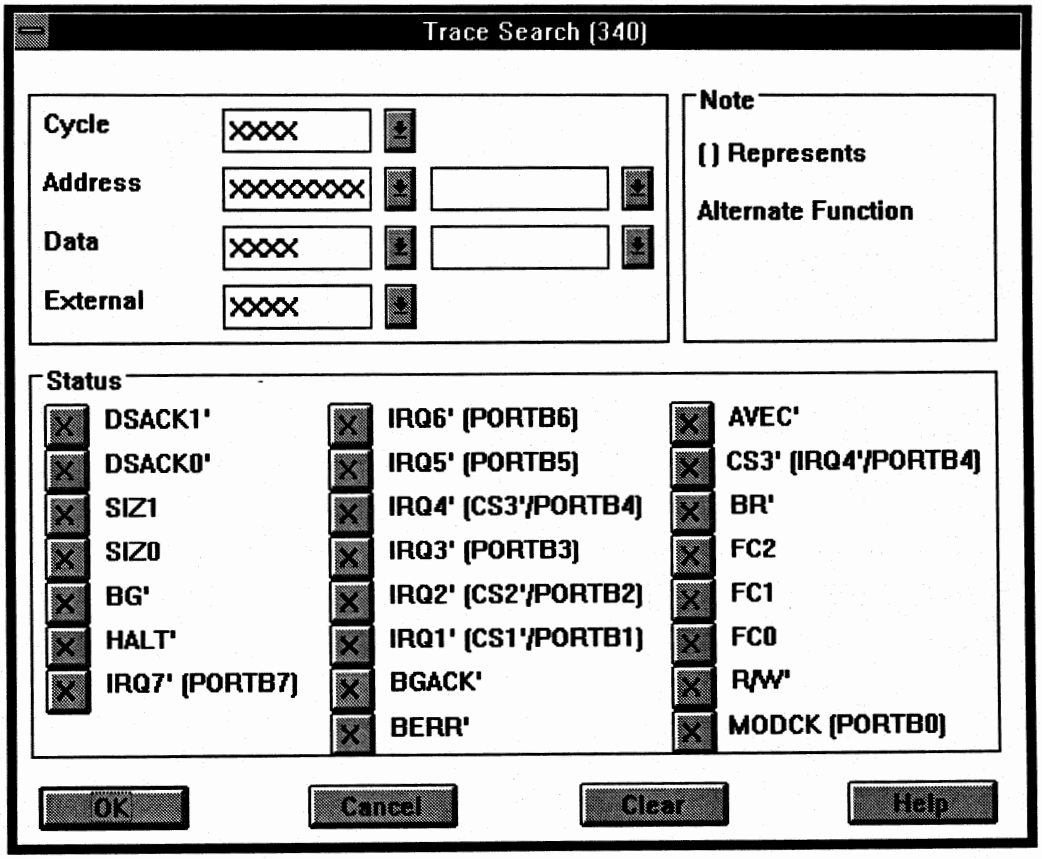
Resuming The Trace

The 'Resume' command can be issued in one of three ways. From the Trace window, you can either choose 'Trace/Resume' or press the 'Resume' button. The 'Resume' command can also be issued from the 'Run Status' window. After the 'Resume' command is given, the emulator will begin looking for the latest breakpoint, trigger point, etc. conditions that were defined during the 'Freeze' operation.

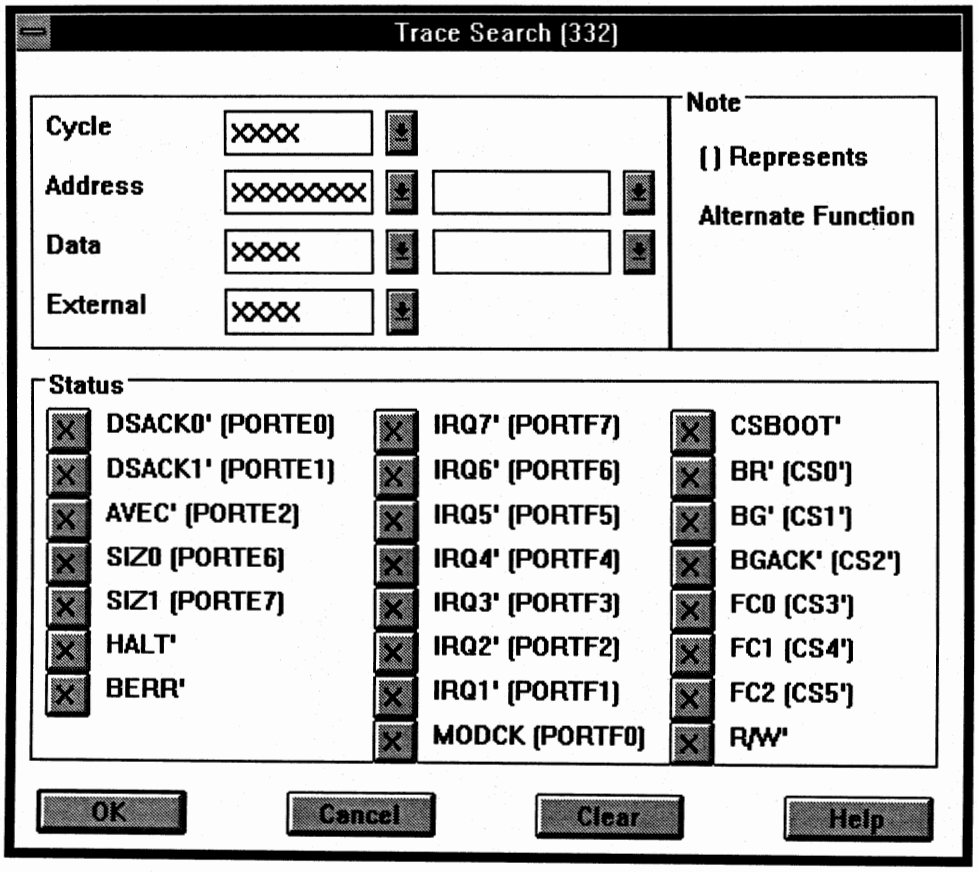
Searching The Trace

You can search the trace buffers for a particular condition. To search the trace buffer, choose "Trace/Find" or press the "Find" button. This will open the Trace Search window. You can then enter the pattern of information to be searched for. The search condition can be any combination of cycle, address, data, external trace bits, and status bits.

The CPU32 Trace Search window has the following format for the 68330/68340:



The CPU32 Trace Search window has the following format for the 68331/68332/68333:



The Cycle, Address, Data, and External values can be changed by selecting the desired field and entering a value for that field. Previously entered values can be obtained by using the scroll box button to reveal and then select the desired value. Symbols can be entered for the Address and Data fields by pressing the right mouse button while the cursor is in one of these fields. The status bit values can be changed by placing the mouse pointer on the desired status bit and pressing the left mouse button. This will toggle through the choices for the status bits. The choices are 0, 1, and 'X' where 'X' is the don't care condition.

After the search function is started, the entire trace buffer will be searched and the first cycle where the search condition is met will be shown at the top of the buffer. To search for the next occurrence of the defined pattern, choose "Trace/Find Next" or press the **Find Next** button.

68330/68340 Trace Search Status Bits

A description of each of the status bits for the 68330/68340 is given below:

HALT':
The HALT' status bit allows you to define a condition to be when a HALT condition occurs (HALT'= 0). The inactive state of this signal is (HALT'= 1). An "X" is the don't care condition which means the status bit will be ignored.

IRQ7' thru IRQ1':
The IRQ7'(PORTB7), IRQ6'(PORTB6), IRQ5'(PORTB5), IRQ4'(CS3'/PORTB4), IRQ3'(PORTB3), IRQ2'(CS2'/PORTB2), and IRQ1'(CS1'/PORTB1) status bits allow you to define a condition to be when an interrupt is occurring (IRQX'= 0). The inactive state of these signals is (IRQX'= 1). An "X" is the don't care condition which means these status bits will be ignored. Note that () is the alternate function depending on how the pin is programmed.

BGACK':
The BGACK' status bit allows you to define a condition to be when an external device assumes bus mastership (BGACK'=0). The inactive state of this signal is (BGACK'= 1). An "X" is the don't care condition which means the status bit will be ignored.

BERR':
The BERR' status bit allows you to define a condition to be when an invalid bus operation is being attempted, or when used with the HALT signal, that the processor should retry the current cycle (BERR'=0). The inactive state of this signal is (BERR'= 1). An "X" is the don't care condition which means the status bit will be ignored.

AVEC':
The AVEC'(CS0') status bit allows you to define a condition to be when the processor generates an automatic vector during an interrupt acknowledge cycle (AVEC'= 0) or when this signal is not active (AVEC'= 1). An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

BG':
The BG' status bit allows you to define a condition to be when a BG' is asserted in response to a BR'. The inactive state of this signal is (BG'= 1). An "X" is the don't care condition which means the status bit will be ignored.

BR':
The BR' status bit allows you to define a condition to be when a BR' is issued. The inactive state of this signal is (BR'= 1). An "X" is the don't care condition which means the status bit will be ignored.

FC0, FC1, and FC2:
The FC0, FC1, and FC2 status bits allow you to define a condition to be when the function codes take on a certain value. An "X" is the don't care condition which means the status bit will be ignored.

R/W':
The R/W' status bit allows you to define a condition to be when a read occurs (R/W'= 1) or when a write occurs (R/W'= 0). An "X" is the don't care condition which means the status bit will be ignored.

DSACK0' and DSACK1':
The DSACK0' and DSACK1' status bits allow you to define a condition to be when each of these signals is active (equal to 0) or not active (equal to 1). An "X" is the don't care condition which means the status bit will be ignored.

SIZ0 and SIZ1:
The SIZ0 and SIZ1 status bits allow you to define a condition to be when each of these signals takes on a certain value. An "X" is the don't care condition which means the status bit will be ignored.

CS3' (IRQ4'/PORTB4):
The CS3' (IRQ4'/PORTB4) status bit allows you to define a condition to be when this chip select takes on a certain value. An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

MODCLK (PORTB0):
The MODCK(PORTB0) status bit allows you to define a condition to be when this signal is active (equal to 1) or not active (equal to 0). An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

68331/68332 Trace Search Status Bits:
A description of each of the status bits for the 68331/68332 is given below:

HALT':
The HALT' status bit allows you to define a condition to be when a HALT condition occurs (HALT'= 0). The inactive state of this signal is (HALT'= 1). An "X" is the don't care condition which means the status bit will be ignored.

IRQ7' thru IRQ1':
The IRQ7'(PORTF7), IRQ6'(PORTF6), IRQ5'(PORTF5), IRQ4'(PORTF4), IRQ3'(PORTF3), IRQ2'(PORTF2), and IRQ1'(PORTF1) status bits allow you to define a condition to be when an interrupt is occurring (IRQX'= 0). The inactive state of these signals is (IRQX'= 1). An "X" is the don't care condition which means these status bits will be ignored. Note that () is the alternate function depending on how the pin is programmed.

BGACK':
The BGACK'(CS2') status bit allows you to define a condition to be when an external device assumes bus mastership (BGACK'=0). The inactive state of this signal is (BGACK'= 1). An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

BERR':
The BERR' status bit allows you to define a condition to be when an invalid bus operation is being attempted, or when used with the HALT signal, that the processor should retry the current cycle (BERR'=0). The inactive state of this signal is (BERR'= 1). An "X" is the don't care condition which means the status bit will be ignored.

AVEC':
The AVEC'(PORTE2) status bit allows you to define a condition to be when the processor generates an automatic vector during an interrupt acknowledge cycle (AVEC'= 0) or when this signal is not active (AVEC'= 1). An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

BG':
The BG'(CS1') status bit allows you to define a condition to be when a BG' is asserted in response to a BR'. The inactive state of this signal is (BG'= 1). An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

BR':
The BR'(CS0') status bit allows you to define a condition to be when a BR' is issued. The inactive state of this signal is (BR'= 1). An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

FC0, FC1, and FC2:
The FC0(CS3'),FC1(CS4'), and FC2(CS5') status bits allow you to define a condition to be when the function codes take on a certain value. An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

R/W':
The R/W' status bit allows you to define a condition to be when a read occurs (R/W'= 1) or when a write occurs (R/W'= 0). An "X" is the don't care condition which means the status bit will be ignored.

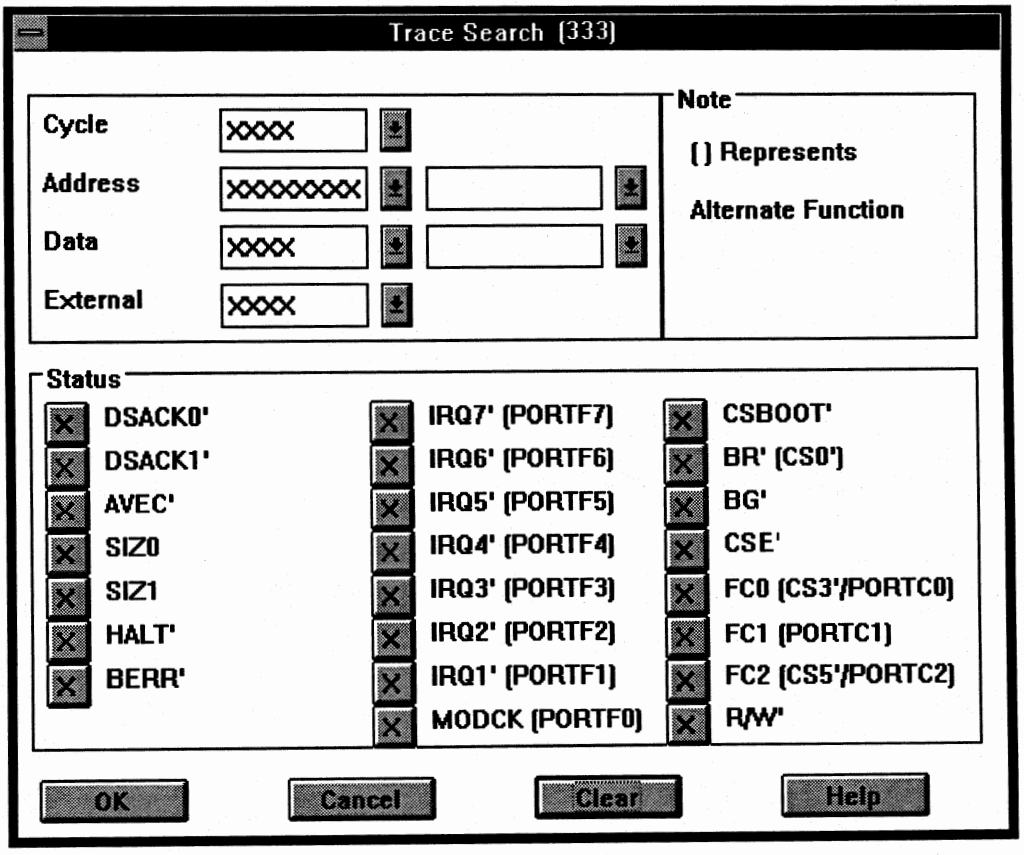
DSACK0' and DSACK1':
The DSACK0'(PORTE0) and DSACK1'(PORTE1) status bits allow you to define a condition to be when each of these signals is active (equal to 0) or not active (equal to 1). An "X" is the don't care condition which means the status bit will be ignored.

SIZ0 and SIZ1:
The SIZ0 (PORTE6) and SIZ1(PORTE7) status bits allow you to define a condition to be when each of these signals takes on a certain value. An "X" is the don't care condition which means the status bit will be ignored.

CSBOOT':
The CSBOOT' status bit allows you to define a condition to be when this chip select takes on a certain value. An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

MODCLK (PORTF0):
The MODCK(PORTF0) status bit allows you to define a condition to be when this signal is active (equal to 1) or not active (equal to 0). An "X" is the don't care condition which means the status bit will be ignored. Note that () is the alternate function depending on how the pin is programmed.

***68333 Trace Search Status Bits*:**
The trace search status bits shown for the 68333 are the same as are shown for the 68332. However, the 68333 does not support some of the alternate functions of these pins. The following is a listing of the status bits and any available alternate functions, shown enclosed in (), that are supported by the emulator for this processor. These status bits correspond to the positions of the status bits shown in the 68332 Trace Search window and should be substituted accordingly:



Notice that BGACK' is not available as a status bit since the emulator must use its alternate function, CSE', to talk to the Port Replacement Unit (PRU) located on the pod. The user's code SHOULD NOT program BGACK' to be active as this will cause the processor to lose communication with the PRU.

Trace Search Window Buttons

A description of the functions of the buttons available in this window are as follows:

0, 1, X

The **0, 1, X** button will toggle through the available choices for the status bits. The status bit values can be changed by placing the mouse pointer on the desired status bit and pressing the left mouse button. This will toggle through the choices for the status bits. The choices are **0**, **1**, and **X** where **X** is the don't care condition.

Ok

The **Ok** button is used to accept the changes made in the search pattern and to begin the search.

Clear

The **Clear** button will set all entries in this window to "X"s (don't care).

Cancel

The **Cancel** button will cancel the changes made in the search pattern and will close the Trace Search window.

Help

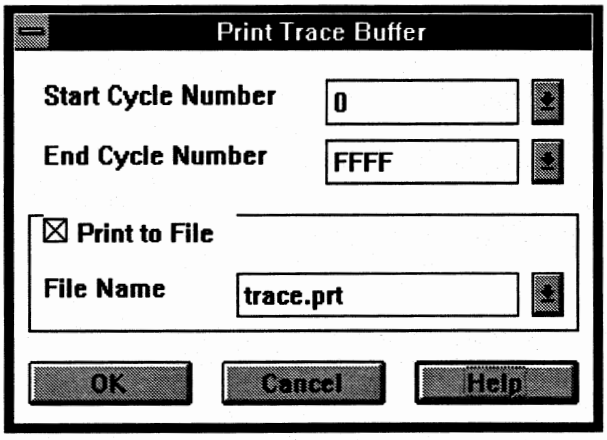
The **Help** button will invoke help for this window.

Scroll Box

The **Scroll Box** button will allow you to select a previously entered value for that field.

Printing The Trace

To print the trace buffer, choose "File/Print". This will open the 'Trace Print' window. The format of this window is shown as follows:



You can define the beginning and ending cycle number which determines how much of the trace information is output. If the 'Print to File' box is checked, you can print the trace information to a file instead of a printer. Simply enter the filename in the 'File Name' field and then press the OK button. If the 'Print to File' box is not checked, the trace output will be do the default printer. To define the printer to be used, choose "File/Printer Setup".

Print Trace Buffer Window Buttons

A description of the functions of the buttons available in this window are as follows:

Ok

The **Ok** button will begin the trace printing operation.

Cancel

The **Cancel** button will abort the trace printing operation.

Help

The **Help** button will invoke help for this window.

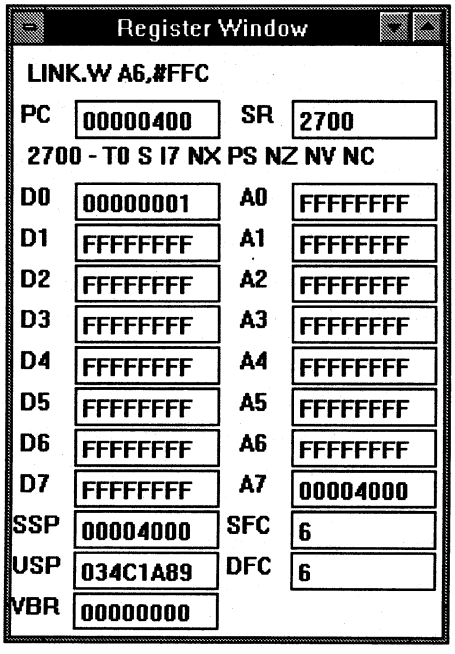
Scroll Box

The **Scroll Box** button will allow you to select a previously entered value for that field.

The CPU32 Register Window

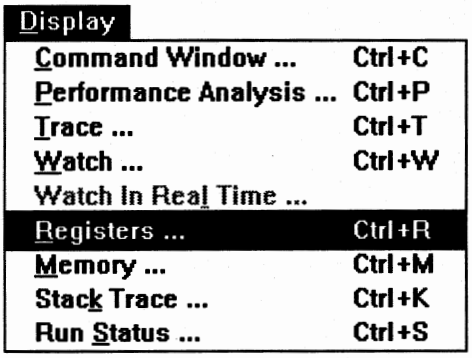
The Register window shows the contents of the processor's registers including the status of the flags and the current disassembly mnemonic being pointed to by the program counter register. The register values are always accurate whenever emulation is not in progress, and are the last known value whenever emulation is in progress. Registers can be altered in the register display window by selecting the desired register data field and then and entering a new value.

The Register window is automatically updated whenever the emulator is single-stepped, a breakpoint occurs, a stop command is executed, or the emulator is reset. The registers shown in the Register window are dependent on the emulator being used. An example CPU32 Register window is shown below:



Opening The Register Window

To open the Register window, choose 'Display/Registers':



An alternate method of opening the Register window is to use the accelerator key combination Ctrl+R.

Modifying Register Values

To modify a register, move the insertion bar to the register to be changed and then press the left mouse button. You can then enter the desired information beginning at the insertion point. To delete the old contents of a register and add a completely new value, click and drag to highlight the entire contents of the register and then enter the new value.

Register Window Contents

The registers shown in the Register window are as follows:

Program Counter Register

The Program Counter is a 32-bit register that contains the address of the next instruction to be executed by the CPU32.

Program Counter Description Field

The Program Counter Description field is the field above the PC field. This field shows the assembled instruction pointed to by the PC and is automatically updated by the emulator. This field cannot be edited.

Status Register

The Status Register is a 16-bit register in which each bit is used to indicate the results of the instruction just executed. The user byte containing the condition codes is the only portion of the Status Register information available at the user privilege level, and it is referenced as the CCR in user programs. In the supervisor privilege level, software can access the full Status Register, including the interrupt priority mask (three bits) as well as additional control bits.

The individual condition codes are also displayed below the hexadecimal SR value. An explanation of each condition code is listed below:

SR Bit	Flag or Condition Code Name	Not Active Code	Active Code
15	Trace Enable T1	T0 or T1	T2 or T3
14	Trace Enable T0	T0 or T2	T1 or T3
13	Super/User State	U	S
12	Not Defined	--	--
11	Not Defined	--	--
10	Int Priority Mask 2		
9	Int Priority Mask 1	I0 through I7	
8	Int Priority Mask 0		
7	Not Defined	--	--
6	Not Defined	--	--
5	Not Defined	--	--
4	Extend	NX	XT
3	Negative	PS	NG
2	Zero	NZ	ZR
1	Overflow	NV	OV
0	Carry	NC	CY

Supervisor Stack Pointer Register

The Supervisor Stack Pointer is a 32-bit register that contains the address of the next free location on the stack if the SSP is the active stack pointer. The SSP is only accessible at the Supervisor privilege level. The SSP value can be modified by placing the cursor on the desired field and entering a new value.

User Stack Pointer Register

The User Stack Pointer is a 32-bit register that contains the address of the next free location on the stack if the USP is the active stack pointer.

Vector Base Register

The Vector Base Register is a 32-bit register that contains the base address of the exception vector table in memory. The displacement of an exception vector is added to the value in this register to access the vector table.

Data Registers D0-D7

Registers D0 - D7 are 32-bit data registers that are used for bit, bit-field, byte, word, long word, and quad word operations. These register values can be modified by placing the cursor on the desired field and entering a new value.

Address Registers A0-A7

Registers A0 - A6 are 32-bit registers that may be used as software stack pointers or base address registers. Register A7 is a register designation that applies to the USP in the user privilege level, and to either the ISP or MSP in the supervisor privilege level.

Alternate Function Code Registers

The Alternate Function Code registers DFC and SFC contain 3-bit function codes. Function codes can be considered extentions of the 32-bit linear address that optionally provide as many as eight 4-gigabyte address spaces. Function codes are automatically generated by the processor to select address spaces for data and program at the user and supervisor privilege levels, and a CPU address space used for processor functions (for example, coprocessor communications). Registers DFC and SFC are used by certain instructions to explicitly specify the function codes for operations.

Index

SYMBOLS

:, 15
:=A, 16
:=C, 16
:=D, 16
:=M, 16
:=S, 17
:module/symbol, 15
:module/symbol=value, 15
:symbol, 15
:symbol*, 15
:symbol=value, 15

A

A (In-Line Assembly), 20
Address Registers A0-A7, 111
Alternate Function Code Registers, 112
Assembly Mode Command, 16
Available CPU32 Commands, 18

B

B (Byte Mode), 13
Base & FC0-2, 6
BM (Byte Mode), 21

C

C (Compare Memory Block), 22
Chip Selects, 6
Clear Symbol Table Command, 16
Clock, 4
CM (Show Status Of Control Board Memory), 24
Command Line Overrides, 13
Command Window, 11
CONFIG (Recall Factory Configuration), 25
CONFIGT (Recall Factory Configuration For Target), 26
Configuration Window, 2
CRL (Load Configuration Registers), 27

D

D (Data Mode), 13
D (Dump Memory Block), 28
Data Registers D0-D7, 111
Default Port Width, 5
Display Current Modes Command, 16
DM (Data Memory Mode), 29
DSACK Source Is, 4
DW (Write Line Active For Target Writes Only), 30

E

E/EN (Enter/Change Memory), 31
EDT (Events Disabled For Temp. Breakpoints), 32
EET (Events Enabled For Temp. Breakpoints), 33
Emulator To Computer Connection, 1
Emulator To Target System Connection, 1
Entering Commands, 12
ET (Show Status For EDT/EET), 34
Event, 73
Event Address Binary, 76
Event Data Binary, 77
Event Status Binary, 78
Event Address, 74
Event Address Range, 74
Event Data, 74
Event Data Range, 74
Event External Binary, 77
Event Pass Count, 75
Event Window, 73
Event Window Binary Fields, 76
Event Window Hex Fields, 73
EW (Write Line Active For All Memory Writes), 35
EXP (Examine/Change Peripheral), 36
EXR (Examine/Change Peripheral Register), 37

F

F (Fill Memory Block), 38
Freeze Trace, 99

G

G (Begin Real-Time Emulation), 39
Graphical Configuration Windows, 9

H

HLD (Hex Listing Disabled), 40
HLE (Hex Listing Enabled), 41

I

I/IN (Input), 42

L

L (List Disassembled Code), 43
L (Long Word Mode), 13
LM (Long Word Display Mode), 44

M

M (Move Memory Block), 45
Memory Configuration, 2
Memory Map, 2
Mixed Mode Command, 16
Modifying Register Values, 110
Module Commands, 16
MT/MTL (Test Block Of Memory), 46

O

O/ON (Output), 47
OFF, 7
Override Example, 14

P

P (Program Mode), 13
PD (Program Wait States For DSACK), 48
PM (Program Memory Mode), 49
Printing The Trace Buffers, 108
Program Counter Description Field, 110
Program Counter Register, 110

R

R (Repeat), 14
RD/RDP/RDX (Read Object File Into Memory), 50
Register Reset Values, 5
Register Window, 109
Register Window Contents, 110
Resume Trace, 99
RP (Reset Processor, Continue Emulation), 52
RS (Reset Processor, Stop Emulation), 53

S

S (Supervisor Mode), 13

Index

Searching The Trace Buffers, 100
SH (Show Status For SHD/SHE), 54
SHD (Serial Handshake Disabled), 55
SHE (Serial Handshake Enabled), 56
SM (Supervisor Memory Mode), 57
Source Code Interface Options, 10
Source Mode Command, 17
SP (Stop Emulation), 58
SR/SRN (Search Block Of Memory), 59
SS (Single Step), 61
ST (Show Emulator Status), 64
Starting SourceGate II, 1
Status Register, 110
Supervisor Stack Pointer Register, 111
Symbol Commands, 15

T

T (Target System), 13
Target System: Br,Halt,Berr,Irqs, 4
Target System: Reset, 4
Terminal Type Configuration Windows, 2
Time Tag Format, 96
Timing Delta, 98
Timing Relative, 96
Trace Buffer Selection, 86
Trace Display Format, 86
Trace Search Window, 100
Trace Status Bits, 89, 92, 95
Trace Window, 83

U

U (User Mode), 13
UM (User Memory Mode), 65
User Stack Pointer Register, 111

V

VASM (View Assembly), 66
Vector Base Register, 111
VER (Display Firmware Version), 67

W

W/WX (Write Memory Contents To File), 68
W (Word Mode), 13
WM (Word Display Mode), 70

X

X (Examine/Change Registers), 71